THESIS

SATELLITE TASKING VIA A TABLET COMPUTER

by

Courtney A. Guy

September 2015

Thesis Advisor:	M. Karpenko
Second Reader:	I. M. Ross

Reissued 3 Mar 2015 with corrected degree

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2015	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE SATELLITE TASKING VIA A TABLET COMPUTER			**5. FUNDING NUMBERS**
6. AUTHOR(S) Guy, Courtney A.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Tablet computing has the potential to reshape the scope of situational awareness. This is because application developers have derived uses for tablet devices that the original inventors did not intend and could not have imagined. One such application is to provide the ability for the warfighter to directly request aerial images from overhead assets, including unmanned aerial vehicles or satellites. Advancements in mobile technology and network connectivity have helped to overcome the challenges of information delivery, but there remains the challenge of real-time information. This thesis examines the concept of tablet-based information requests for real-time satellite tasking. As a proof-of-concept, a tablet-based application is developed that enables the user to task a satellite system by interacting with a map. Requests are sent and processed by a server application and are then routed to the appropriate asset. Real-time response to the request is emulated using a detailed simulation model of a control moment gyroscope actuated spacecraft. Simulated images and spacecraft attitude errors are used to mimic the data collection process. This information is uploaded to the server for retrieval by the tablet application, thereby completing the request cycle and demonstrating the feasibility of remote satellite tasking using a tablet computer.

14. SUBJECT TERMS tablet applications, Xcode, Objective C, Sencha Touch 2, Cordova, tablet application development, satellite tasking, imaging satellites, spacecraft model, satellite scheduling, MUOS, Kestrel Eye, GeoEye, WorldView, Commercial Imagery Team, imaging assets, ORS, Operationally Responsive Satellites			**15. NUMBER OF PAGES** 115
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

SATELLITE TASKING VIA A TABLET COMPUTER

Courtney A. Guy

B.S., New Jersey Institute of Technology, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2015

Author:	Courtney A. Guy
Approved by:	M. Karpenko, PhD Thesis Advisor
	I. M. Ross, PhD Second Reader
	Garth V. Hobson, PhD Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Tablet computing has the potential to reshape the scope of situational awareness. This is because application developers have derived uses for tablet devices that the original inventors did not intend and could not have imagined. One such application is to provide the ability for the warfighter to directly request aerial images from overhead assets, including unmanned aerial vehicles or satellites. Advancements in mobile technology and network connectivity have helped to overcome the challenges of information delivery, but there remains the challenge of real-time information. This thesis examines the concept of tablet-based information requests for real-time satellite tasking. As a proof-of-concept, a tablet-based application is developed that enables the user to task a simulated satellite system by interacting with a map. Requests are sent and processed by a server application and are then routed to the appropriate asset. Real-time response to the request is emulated using a detailed simulation model of a control moment gyroscope actuated spacecraft, which then provides simulated images to mimic the data collection process. This imagery product is uploaded to the server for retrieval by the tablet application, thereby completing the request cycle and demonstrating the feasibility of remote satellite tasking using a tablet computer.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ADCS	Attitude Determination and Control System
API	Application Programming Interface
ARSTRAT	Army Forces Strategic Command
BRITE	Broadcast Request Imagery Technology Experiment
CIT	Commercial Imagery Team
CMG	Control Moment Gyroscope
DoD	Department of Defense
EVR2EST	Eagle Vision/ROVER Responsive Exploitation of Space Products for Tactical Use
GEOINT	Geospatial Intelligence
GPS	Global Positioning System
HCI	Human Computer Interaction
ISR	Intelligence Surveillance Reconnaissance
IRU	Inertial Reference Unit
MILSAT	Military Satellite
MUOS	Mobile User Objective System
NGA	National Geospatial Intelligence Agency
NGO	Non-Governmental Organization
NIMA	National Imaging and Mapping Agency
NPS	Naval Postgraduate School
NSG	National System of Geospatial Intelligence
SATCOM	Satellite Communications
SIPRNET	Secret Internet Protocol Routed Network

SMDC	Space and Missile Defense Command (US Army)
STK	Satellite Tool Kit
TASS	Three-Axis Satellite Simulator
TINYSCOPE	Tactical Imaging Nano-satellite Yielding Small-Cost Operations for Persistent Earth Coverage
UDID	Unique Device Identifier
UI	User Interface
URL	Uniform Resource Locator
USASMDC	United States Army Space and Missile Defense Command
WGS84	World Geodetic System of 1984

NOMENCLATURE

m^{B}	mass of body B
$D^A(*)$	rotational derivative of $*$ with respect to frame A
v_B^A	velocity vector of point B with respect to frame A
$\mathbf{H}_s$	angular momentum vector
$\dot{\mathbf{H}}_s$	rate of change of angular momentum vector
$\mathbf{T}_{\mathrm{external}}$	external torque vector
$\boldsymbol{\omega}$	spacecraft angular velocity vector
$\dot{\boldsymbol{\omega}}$	spacecraft angular acceleration vector
$\mathbf{h}$	total CMG momentum vector
$\dot{\mathbf{h}}$	total CMG momentum rate vector
$\boldsymbol{\delta}$	CMG gimbal angles
$\dot{\boldsymbol{\delta}}$	CMG gimbal rates
μ	gravitational parameter ($\mathrm{m}^3/\mathrm{s}^2$)
h_{orb}	orbit altitude
R_E	Earth radius
β	skew angle of CMGs
$\phi_{s,f}$	latitude, s-subscript for start, f-subscript for finish
$\lambda_{s,f}$	longitude, s-subscript for start, f-subscript for finish
θ	roll angle

φ	pitch angle
ψ	yaw angle

ACKNOWLEDGMENTS

Thank you to Harry Wong, my manager at Boeing, for his recommendation to this amazing program and experience at NPS. Thank you to the executives of the Boeing Company who provided me this incredible opportunity to study at the Naval Postgraduate School.

Thank you to Dr. I. M. Ross and Dr. M. Karpenko for outlining available thesis topics and for your enthusiasm to be thesis advisors. The concern for my limited tenure at Naval Postgraduate School and shared interest in my success made all the difference to me. Thanks Dr. I. M. Ross for advising me academically and professionally, and for your support in the development of the application and spacecraft model. Thank you to Dr. M. Karpenko for keeping me on track but yet allowing me those days of gloating after a breakthrough.

Thank you to Dr. T. Sands for his help with the spacecraft model. Thank you for your patience in explaining coordinate transformation systems and spacecraft disturbance equations multiple times. I learned so many Simulink fixes and from you, I’m sure there is enough for a book or at least a Simulink for Dummies book. There is not one published yet, I checked.

Thank you to my classmates for their moral support in the Space Cave. Thank you especially to Reid Smythe for all the MATLAB assistance, patience and calming presence.

Thank you to Nick Busey for his Xcode development suggestions, and assistance in selecting and setting up WordPress on the Amazon EC3 server. Thank you to Jerry Tung for his Xcode debugging assistance and introduction to Stack Overflow. Thanks to anonymous forum responders for answering my 3am-questions and for the code examples on GitHub.

THIS PAGE INTENTIONALLY LEFT BLANK

I. BACKGROUND

A. MOTIVATION

Recent advancements in mobile technology and network connectivity have helped to overcome some of the challenges of information delivery and allow for more information to be disseminated quickly. Furthermore, large amounts of data and computation power are now portable as a consequence of the miniaturization of hardware. The convergence of these two separate areas of progress has made tablet computers and smart phones viable. These devices have changed society primarily because the two primary mobile operating systems (Andriod from Google and iOS from Apple) allow any developer to create applications (Mian, Teixeira and Koskivaara 2011). By allowing application development on the mobile platform, devices like the iPhone and iPad can be used in ways the original inventors could not have imagined. When a challenge in everyday life arises, the question is often posed, "Is there an app for that?" There are numerous applications available for occupational efficiency, personal organization, and hobbies. For example, while staring at the night sky, there are several iPhone applications that allow stargazers to point their device in a particular direction and view a star chart identifying the planets and constellations. *SkyView* and *The Night Sky* are two apps currently available from the iTunes store that provide this function. For those more interested in satellites than stars, there are applications available that predict when a satellite, like ISS, or a solar flare from Iridium can be viewed with the naked eye. One app is named *GoSatWatch* and another is *ISS Dectector Satellite Tracker*. With these examples in mind, it is easy to imagine many other applications for tablets that could aid the warfighter or emergency personnel in retrieving timely and accurate information. This thesis examines one such application: the concept of direct satellite tasking within the context of satellite imagery. If a tablet application can request and receive real-time or near real-time imagery from a satellite, it will significantly benefit the warfighter or emergency personnel by providing the time-critical information needed for the success of their missions.

The importance of overhead imagery has been known for decades. Ever since the U2 reconnaissance missions and the Corona program in the 1960s aerial images have provided decision-makers key information about an adversary’s capabilities and movements (Maathuisa and van Genderena 2004). More recently, during operation Enduring Freedom a prototype system called BRITE (Broadcast Request Imagery Technology Experiment) enabled operators to view satellite imagery in near-real time. The BRITE system was operated on a laptop computer and allowed ground forces to relay the coordinates of objects of interest (Lambeth 2001, 276). The laptop-based BRITE system disseminated critical NGA GEOINT data via Military Satellite (MILSAT) Secret Internet Protocol Routed Network (SIPRNET) to communications-limited tactical users worldwide (Office of Geospatial-Intelligence Management 2006). Although this prototype system was useful in providing time critical information, it could have been more valuable if deployed on a tablet platform. Tablets are more portable and usable; their longer battery life means they can be transported farther from a power source (Kukulska-Hulme and Traxler 2005, 45). Their small form factor can be easily carried around and used without the flat surface required to use a laptop. Additionally, it is more natural to pass around a tablet to share information vice spinning around a laptop (Gillett 2012). There are aspects of BRITE that can be strengthened to support the warfighter’s information requirements. For example, the BRITE system requires either that the laptop have a line of site connection to a UHF radio or a Wi-Fi connection through a similar link (Air Land Sea Application Center 2004). A tactical user would rely on that additional link to make the system useful, and also need to transport the laptop. The BRITE system capabilities demonstrate that there is the infrastructure to support the direct tasking of satellites by the warfighter. The concept of directly tasking a satellite via a tablet could improve upon the BRITE system concept by allowing the user a direct connection to the satellite, and to task the satellite from a more practical device.

From the time when BRITE was first deployed until now, there has been a congressional mandate to make existing and future space assets more operationally responsive. This congressional mandate prompted the creation of the Operationally Responsive Space Office (National Security Space Office 2007). The ORS Office activities are divided into three tiers as seen in Figure 1. The first tier involves rapidly exploiting existing capabilities that are net-centric with an open architecture. Replenishing, augmenting and rebuilding with exiting capabilities are tier two approaches to enhancing space responsiveness, and tier three involves developing new technologies.

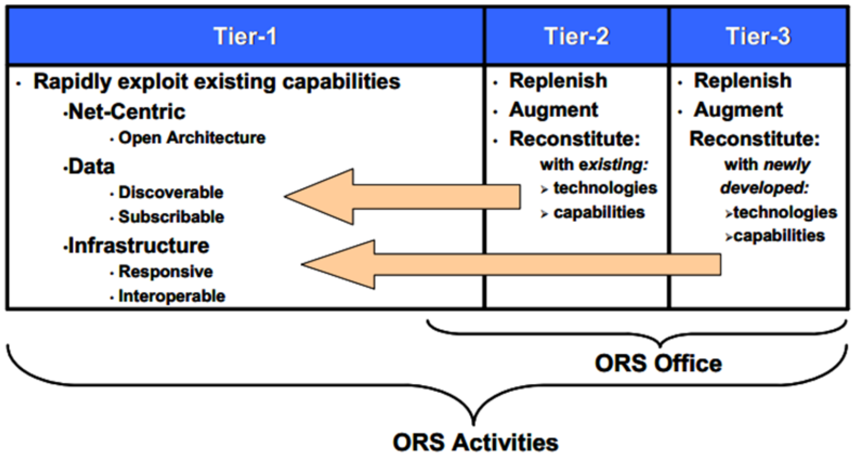


Figure 1. Tiered approach to enhance responsiveness of space capabilities (from National Security Space Office 2007).

The ORS office tiered approach and stated needs for the tactical use of space power provided further motivation for the direct tasking concept (McLaughlin 2007).

B. OVERVIEW

It is important to note that there is a difference between commercial satellite imagery that most people are familiar with and the imagery that the warfighter or emergency personnel require. Google Maps-based applications allow the user to search for a location or generate directions from commercial satellite imagery; there are even advanced settings for downloading the images. This information, although incredibly useful for the layperson, is not as useful to the tactical user. This is because the application does not offer information on the image age and there is no guarantee of accuracy of the image (its location, and visibility). The warfighter and civilian emergency personnel require recently acquired images and need guaranteed and precise information on the time and location of the image. In a statement by Mr. Novak, a firefighter and employee from USASMDC/ ARSTRAT:

> It definitely helps out having that imagery and having that oversight for planners and emergency managers throughout the nation. To have that near-real-time information and then having the updates come in can be the difference between life and death. (Cutshaw, SMDC employee helps nation prepare for emergencies 2012)

Mr. Novak's statement emphasizes the importance for space-based imagery for tactile decision making and the importance of real-time information. The difference between the commercial imagery database available for public consumption and space-based imagery needed for tactical decisions is near-real-time availability and guaranteed data.

There are numerous assets available that could be integrated into a tablet tasking application and network. These assets encompass not only the commercial imaging satellites mentioned earlier, but also include government systems like Airborne Intelligence Surveillance Reconnaissance (ISR) such as Global Hawk and Predator. The imaging hardware is not a solution by itself and requires ground stations, distribution networks, high-bandwidth data communications and worldwide communication coverage. Nonetheless, it is possible to incorporate these assets into a tablet tasking paradigm.

Considering the available imaging assets and communication architectures an overview of how a tablet application could be integrated is depicted in Figure 2. For example, the tablet can directly send requests through a communication satellite such as MUOS (Oetting and Jen 2011). Alternatively, the tablet could transmit directly to an overhead imaging asset through a portable SATCOM terminal like the SurfBeam portable terminal (ViaSat 2012) or similar. Through a server, the commands could be routed from a ground station to the imaging asset, and the resulting images downlinked to its associated ground station. From the ground station, the images can be routed back to the server and made available for retrieval by the tablet.

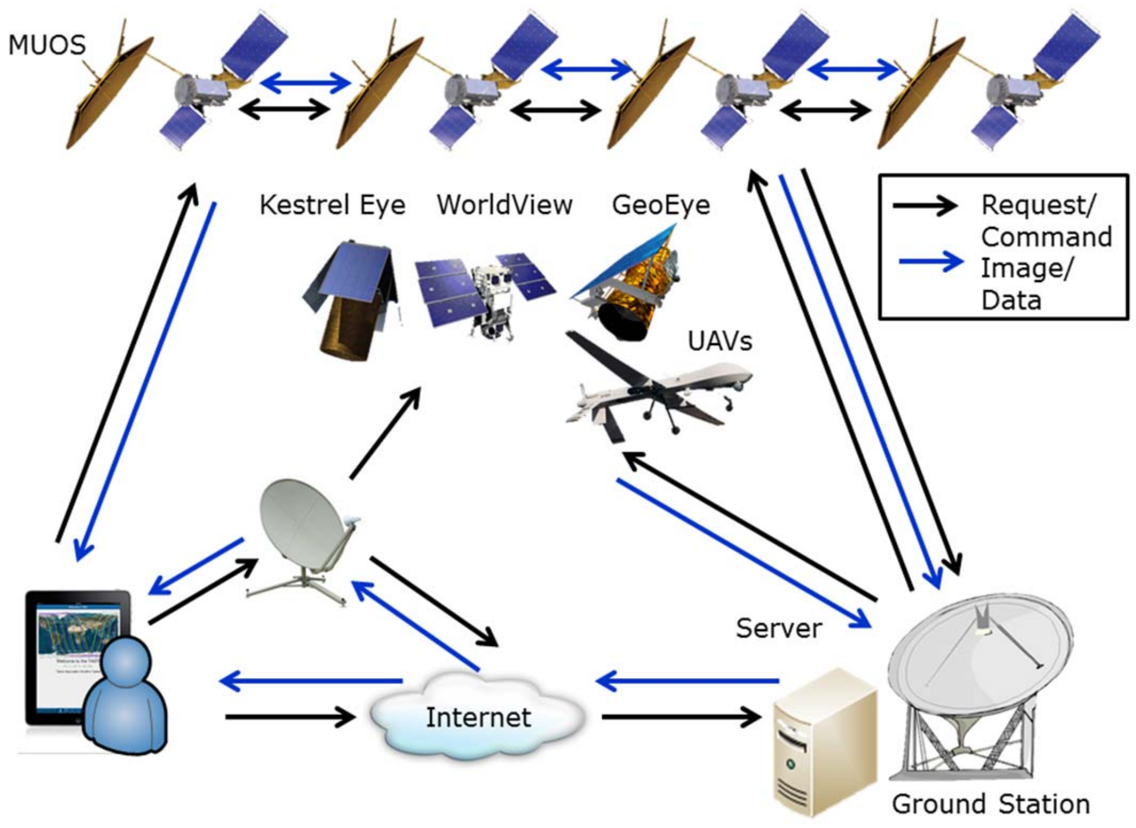


Figure 2. Possible tablet integration with existing infrastructure.

The following sections will provide an overview of the current distribution networks, current commercial imaging resources, future imaging assets and communication architectures that are available to support the architecture of Figure 2. The intent is to provide background for the notion that the tablet tasking concept could conceivably be integrated into existing infrastructure.

C. IMAGERY TASKING AND DISTRIBUTION

The need for space-based imagery on demand became apparent during Desert Storm when U.S. forces had to wait four to six to weeks for the processing of images and delivery to theater (Hartmetz 2001). Since then, programs such as Eagle Vision and other commercial companies have worked to provide time-critical space-based imagery quickly. The key commercial provider of satellite imagery in the United States, DigitalGlobe, has many partners that can task their satellites for a particular target or area (Crampton 2011). These partners use the satellite imagery for geospatial intelligence, government and business planning, navigation and defense purposes.

Eagle Vision is the compilation of several systems including Eagle Vision I, National Eagle, Eagle Vision II and others that collect images from national and commercial assets and also processes the images (Hartmetz 2001). Commercial imagery, as opposed to imagery from National Assets, improves cycle time to users because it is unclassified and immediately shareable. Commercial imagery can be marked unclassified and/or limited distribution and can therefore be easily released to Host Nation personnel, coalition forces, government agencies and NGOs. The Commercial Imagery Team (CIT), part of the U.S. Army Space and Missile Defense Command, advertises that their delivery time for commercial imagery is hours after the satellite has collected the information (Cutshaw, SMDC employee helps nation prepare for emergencies 2012). The same team also recently deployed the Eagle Vision/ROVER Responsive Exploitation of Space Products for Tactical Use (EVR2EST) system. EVR2EST is a system that distributes space-based imagery and radar products generated by the Eagle Vision program. The Eagle Vision program is comprised of commercial imagery ground stations that can downlink imagery directly from satellites and process the images at the same site. Then, EVR2EST is used by civilian emergency personnel and first responders to process, web-optimize, and share information with federal, state and local emergency managers (Cutshaw, SMDC employee helps nation prepare for emergencies 2012).

During the wildfires in Colorado (June 2012), EVR2EST was used to distributed approximately 37 square miles of satellite imagery to the Colorado fire response authorities. The images were distributed through a website to local agencies and firefighters (Cutshaw, EVR2EST helps firefighters during wildfires 2012). If a tablet application could be integrated into the EVR2EST system, users would be able to task very specific areas of interest and have the information available in the field, allowing for quicker updates on the situation.

The National Geospatial Intelligence Agency (NGA) has several programs that aim to provide access and dissemination of imagery including the programs Eagle Vision and BRITE programs previously mentioned. The programs and their focus are listed in Table 1.

Program	Focus
Broadcast-Request Imagery Technology Environment (BRITE)	NSG program that disseminates critical NGA GEOINT data via Military Satellite (MILSAT)/Secret Internet Protocol Routed Network (SIPRNET) to communications-limited tactical users worldwide.
Command Information Libraries (CIL)	Intermediate image library between the NIL and IPL; at command and agency locations.
Commercial Remote Sensing (CRS)	Ground Receiving/Processing Stations Facilities (e.g., Eagle Vision) that generate actionable GEOINT from CRS data. These stations enhance the operational utility of CRS data to operational commanders.
Distributed Common Ground/Surface System (DCGS)	Family of systems designed to provide airborne system-derived, multi-intelligence discipline, ISR task, post, process and use capabilities at the theater and tactical levels.
Image Product Libraries (IPL)	Scaleable, deployable libraries below Command Information Libraries in complexity and capacity.
Information Dissemination Services-Direct Delivery (IDS-D)	NSG program that disseminates time-dominant/time-critical and near-real time data to operational users worldwide. Also sends National Technical Means (NTM) data directly to the NIL for long-term storage.
MC&G Information Library (MCGIL)	Mapping, Charting, and Geodesy Information Library.
National Information Library (NIL)	Central repository of national, tactical, and commercial imagery, imagery products, geospatial information, video and metadata.
Unclassified National Imagery Library (UNIL)	Archive and dissemination of commercial imagery. Will eventually replace the Commercial Satellite Imagery Library (CSIL) as the hub for this activity.
Web-based Access and Retrieval Portal (WARP)	NSG program that provides discovery, access and dissemination of NTM, commercial, airborne, geospatial intelligence products from the NGA Gateway, and a variety of specially tailored products to registered operational users worldwide over Joint Worldwide Intelligence Communications System (JWICS), SIPRNET and the internet.

Table 1. NSG capabilities for accessing, disseminating and archiving GEOINT information (Office of Geospatial-Intelligence Management 2006).

A tablet application could be integrated with the existing capabilities of the NGA programs listed. These programs are buried within the internet, server and ground station depictions seen in Figure 2. The following section describes existing imaging assets and how they could be integrated with the concept of a tablet tasking application.

D. IMAGING ASSETS

The Department of Defense and its subsidiary agencies have dedicated assets for earth imaging but also heavily utilize commercial imagery. Commercial imagery is significant in that it has the potential to deliver timely information to soldiers in the field because of its unclassified nature. The importance of timely commercial imagery to military operations was recognized in a *Defense Daily* article from 1999. The author of the article opens with "High resolution commercial imagery offers the possible benefit of providing commanders in the field more timely battlefield awareness, a capability that could drive considerable demand from the military" (Commercial imagery to provide commanders rapid pictures 1999).

Since 1999, commercial imagery has gained even more prominence. In 2002, the Senate Armed Services Committee (SASC) and House Armed Services Committee (HASC) directed the DoD and the intelligence agencies to make more efficient use of the commercial imagery available (Gildea 2002). The push by lawmakers and the National Imaging and Mapping Agency (NIMA) to use more commercial imagery for over the past decade has supported the growth of the commercial imagery industry (Eisler 2008). Companies like DigitalGlobe have benefitted from government contracts. Since DigitalGlobe's merger with GeoEye in January of 2013, DigitalGlobe reports that half its revenue is government derived (Crampton 2011). A large portion of this revenue comes from the EnhancedView contract awarded before the merger to both GeoEye and DigitalGlobe from the National Geospatial-Intelligence Agency (NGA). The EnhancedView contract was originally a ten-year contract worth over $7.3 billion. The contract is significant to the commercial satellite industry because it was in response the failure of Future Imagery Architecture (FIA) program and shifted budget away from government procured satellites and gave it to commercial imaging companies. This already intertwined relationship between the government and commercial imagery supports the idea that these assets could be used in a direct tasking network for the warfighter. An overview of this merged commercial imagery company is provided as background for current assets that could be used in a direct tasking network.

1. DigitalGlobe Constellation

The DigitalGlobe, Inc. (formerly EarthWatch, incorporated in 2002) constellation now consists of six satellites. In 2013, DigitalGlobe merged with GeoEye and added two satellites to its fleet. These six satellites provide commercial high-resolution earth imagery to defense and intelligence agencies, civil agencies, oil and gas companies, academic and research institutions (DigitalGlobe 2015). The six satellites are WorldView-3 launched in 2014, IKONOS, GeoEye-1, WorldView-2, WorldView-1, and QuickBird.

a. WorldView-3

WorldView-3 launched in 2014 to a 617 km altitude. The sensors on board include an imager able to provide 31cm panchromatic resolution, 1.24 meter multispectral resolution, 3.7 meter short wave infrared resolution, and 30 m CAVIS resolution. CAVIS (Clouds, Aerosols, Vapors, Ice and Snow) monitors the atmospheric conditions to provide correction data for images taken through non-ideal atmospheric conditions, like cloud cover (DigitalGlobe 2015).

b. IKONOS

IKONOS was launched by Space Imaging in 1999 and was the world’s first one-meter resolution commercial Earth imaging satellite (Gildea 2002). It is also capable of 3.2 m multispectral near-infrared imagery. The orbit is 681 km in a 98.1 sun synchronous orbit (Satellite Imaging Corporation 2015).

c. GeoEye 1

GeoEye 1 launched in June of 2008 into a 668 kilometer sun-synchronous orbit. The satellite has three camera modes including simultaneous panchromatic and multispectral, panchromatic and multispectral. Image resolution is declared to be 0.41 meter in the panchromatic mode and 1.65 meters in the multispectral mode. A single scene is 225 square kilometers and nominal swath width is 15.2 kilometers (GeoEye 2011).

d. WorldView-2

WorldView-2 was launched in October of 2009 from Vandenburg Air Force Base. The orbit is sun-synchronous, with an altitude of 770 km. The satellite has a panchromatic sensor and eight multispectral sensor bands. The attitude determination and control system uses control moment gyros, star trackers, a solid state IRU and GPS (DigitalGlobe 2015).

e. Worldview-1

In 2007, WorldView-1 was launched from Vandenburg Air Force Base. It is also in a sun-synchronous, 495 km altitude orbit. At nadir the ground sample distance GSD is 50 cm, with a swath width of 17.7 km. The attitude determination and control system uses control moment gyros, star trackers, a solid state IRU and GPS (DigitalGlobe 2015).

f. QuickBird

DigitalGlobe’s Quickbird satellite offers sub-meter resolution imagery (65 cm panchromatic at nadir), and high geolocational accuracy. The swath width of the sensor is 18.0 km and supports a multitude of geospatial applications. According to the company’s website Quickbird is still operating at an altitude of 400 km in a gradual decent and its mission will cease once it reaches 300 km (DigitalGlobe 2015).

2. Other Assets

Aside from the assets mentioned above, there is also commercially available radar (Canada’s Radarsat), LIDAR and thermal remote sensing systems on airborne and spaceborne platforms. These assets plus the multispectral, and hyperspectral assets offer important sources of quality and timely information that can serve the operational needs of the warfighter (Birk et al. 2003). If these assets and their availabilities could be centrally scheduled, a tablet application with the ability to task any of these assets could be an extremely powerful tool in providing exceptional situational awareness. Granted, centrally scheduling all these assets is a lofty goal; however, if only a few of these assets provided direct tasking capabilities to support critical missions, warfighters and emergency personnel could greatly improve their situational awareness.

E. FUTURE IMAGING ASSETS

1. Kestrel Eye

The U.S. Army Kestrel Eye 1 Tactical Imaging Spacecraft is slated to launch in late 2015 on a Falcon 9 as a Spaceflight Secondary Payload System (Vance 2014). The Kestrel Eye spacecraft is said to weigh approximately 14 kg and is classified as a nano-imaging satellite. Steve Fujikawa of IntelliTech Microsystems Inc, says "Kestrel Eye will be taskable directly by the warfighter under fire and transmit 1.5-meter resolution images directly to his backpackable ground station" (KMI Media Group 2009). The full constellation of 30 satellites, once launched, will provide global coverage.

The Kestrel Eye architecture would be a great platform to integrate with a tablet application for several reasons. Not only does the overarching goal of Kestrel Eye to provide tactical level space asset control match the objective of tablet tasking, a tablet application to task a satellite conforms to the operational concept released by SMDC. The operational concept for Kestrel Eye includes a user selecting a point on a satellite ground trace of a world map, and receiving the requested images via a data relay network accessible by the warfighter (USASMDC/ARSTRAT/Public Affairs Office 2010).

2. TINYSCOPE

TINYSCOPE stands for Tactical Imaging Nanosat Yielding Small-Cost Operations for Persistent Earth-coverage. It is a project at Naval Postgraduate School intended to demonstrate the utility of small spacecraft for tactical imagery (Blocker 2008). The mission of the project is to provide a densely populated constellation of low-cost imaging satellites, with a revisit rate of less than 30 minutes to provide tactically relevant and useful information to warfighter (Litton 2009). A tablet application that could task the satellites directly would integrate well with the TINYSCOPE.

3. SeeMe

The SeeMe program, Space Enabled Effects for Military Engagements, aims to provide temporary low cost satellites (less than $500,000) that can be launched quickly to support military operations (Vance 2014). The SeeMe satellites will provide on-demand imagery to the lowest-echelon warfighter in the field (Keller 2012). The goal of the constellation is to provide persistent coverage with no coverage gaps greater than 90 minutes. The Defense Advanced Research Projects Agency, DARPA, awarded Raytheon $1.5 million in December of 2014 for the development of these satellites (Raytheon 2015). A tablet application that could task these satellites directly would integrate well with this DARPA project and help meet the project goals.

F. COMMUNICATION ASSETS

There are a number of satellite networks that could conceivably be used to route the requests from a ground station to an available imaging asset. Platforms like Globalstar sell a la carte packages for voice, data, and short messaging services. A Globalstar satellite and voice module could be conceivably integrated into a tablet and then the device could transmit short request messages through the Globalstar network (Globalstar 2011). It is more interesting, however, to investigate the possibility of using an existing platform to accept the request directly from a tablet-like device and have the ability to either forward it directly to an imaging asset or over to a master scheduling facility. The MUOS platform seems innately adept at being able to support satellite tasking via a tablet-like device.

The MUOS satellite architecture is designed to support handheld terminals, like tables. The MUOS satellites have a multibeam antenna and a 14-meter reflector for transmitting and receiving MUOS UHF WCDMA signals. A diagram of the MUOS architecture is given in Figure 3. The multibeam antenna forms 16 beams for higher antenna gains to allow for handheld, and lower transmit power terminals (Oetting and Jen 2011). If future development of MUOS-capable terminals incorporated a tablet computer, there are endless possibilities for capability improvements. One possible improvement is the use of a tablet application to request an image. A user with a combined tablet computer and MUOS-capable terminal could use an application to select a target on a map and submit a request directly to the CIT or other commercial imagery provider through the MUOS architecture.

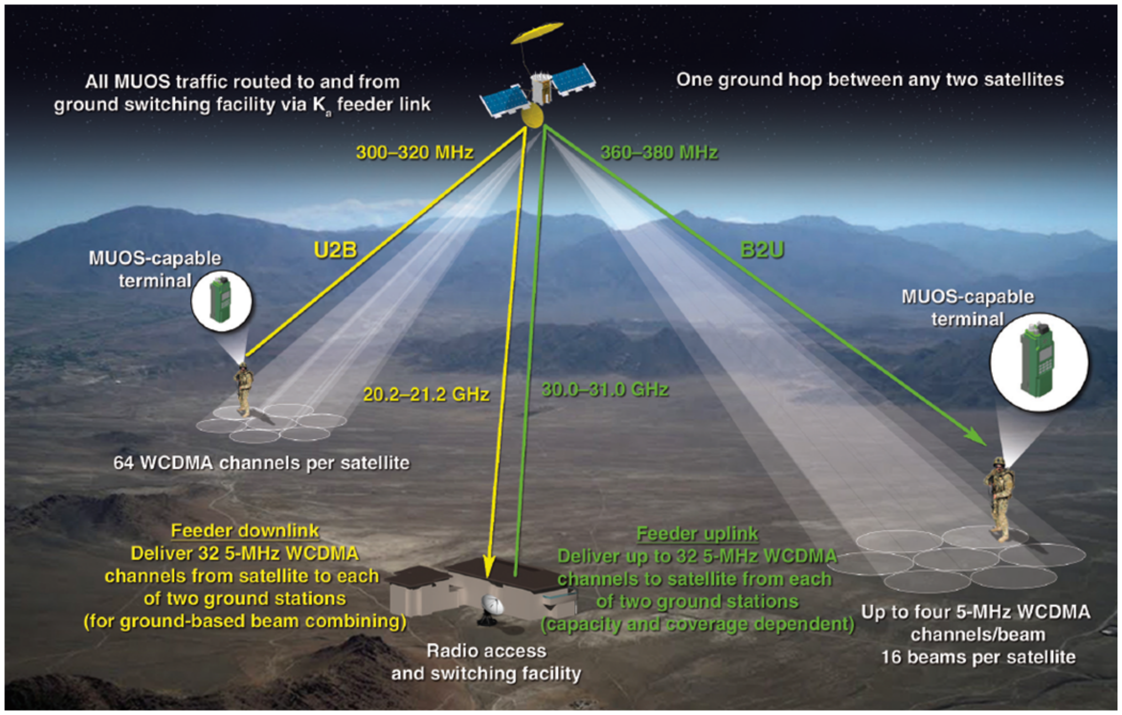

All MUOS traffic routed to and from ground switching facility via K_a feeder link
One ground hop between any two satellites
300–320 MHz
360–380 MHz
U2B
B2U
MUOS-capable terminal
MUOS-capable terminal
20.2–21.2 GHz
30.0–31.0 GHz
64 WCDMA channels per satellite
Feeder downlink
Deliver 32 5-MHz WCDMA channels from satellite to each of two ground stations (for ground-based beam combining)
Feeder uplink
Deliver up to 32 5-MHz WCDMA channels to satellite from each of two ground stations (capacity and coverage dependent)
Up to four 5-MHz WCDMA channels/beam
16 beams per satellite
Radio access and switching facility

Figure 3. MUOS architecture (from Oetting and Jen 2011).

If the tablet application could interface with the MUOS-capable terminal, the user would be able to send a request directly to a MUOS satellite network. From the MUOS network, the task could be sent directly to a small satellite like TINYSCOPE. The advantage of using the MUOS architecture is that the ground terminals do not need to have high-powered transmitters, and the MUOS constellation provides near-world-wide coverage. The MUOS satellites have sensitive receivers that are not necessarily feasible on small satellites due to the strict mass and power limitations. If communication with small satellites was feasible from the tablet, then the limitation would be on available communication windows. The near-world-wide coverage of MUOS means that the user does not need to wait for a satellite to pass overhead to transmit the task or receive the data (assuming other internet connections are not available), and that the MUOS capable terminal does not need a high powered transmitter. This implies that a MUOS capable-terminal could conceivably be integrated into a tablet with the tasking application.

G. THESIS OUTLINE AND SCOPE

The objective of this thesis is to examine the concept of directly tasking a satellite from a tablet computer. The thesis reviews existing and future architectures relating to the direct tasking concept. Programs with similar missions of providing warfighters and emergency personnel with on-demand overhead imagery are identified in this thesis for possible future integration.

The proposed tablet application in this thesis, combined with the knowledge of the existing BRITE system, implies a clean fit into the Tier-1 ORS activities discussed earlier. The existing dissemination programs, commercial imagery assets, and communication networks comprise the existing infrastructure that was considered for this proof of concept. National assets were not discussed because the distribution of commercial imagery can be done more quickly. There is no time lost in approving and sanitizing the imagery. The next chapter of the thesis describes the tablet application that was developed, its platform and implementation. It also discusses some of the design choices and trades that were made. A cursory study of human computer interaction was conducted to help start the development of the application on the right track to meet the end requirement of creating a user-friendly application and interface.

It is impractical to test the tasking ability of a tablet application without first extensively testing the concept and application on the ground. Satellites are very expensive assets and stakeholders typically do not take lightly to anyone “experimenting” on their systems. Therefore, the ability of the developed tablet application to task a simulated spacecraft is examined. Chapter III described the simulated spacecraft and chapter IV presents a sample scenario of the application tasking the simulator.

The last chapter of this thesis addresses the next steps required to fully automate the tasking process, such as testing the application interface using a ground test bed.

There are many suggestions for future work addressed in relation to making the application properly fit the needs of the warfighter and emergency personnel.

THIS PAGE INTENTIONALLY LEFT BLANK

II. TABLET TASKING APPLICATION DEVELOPMENT

A. INTRODUCTION

This chapter describes the approach taken for building a tablet application for tasking a satellite and discusses trades made in the development of the application. During the development process it became clear that there exist many possibilities for implementing a suitable application. There are different ways for tablets to transmit information, including Bluetooth, Wi-Fi, and CDMA. Additionally, there are innumerable ways to design a tablet application. Consequently, the application developed is simply one instantiation of how to realize the idea of directly tasking a satellite. The design choices for the developed application were made by the author who is a satellite systems engineer, and not by a computer programmer or human-computer interaction specialist. Significant research was conducted to bridge the existing satellite knowledge base with programming and concepts from human-computer interaction, but research beyond the scope of this work is likely needed to create the best application for the user. Nonetheless, the application as designed, is able to transmit the required information to task a satellite. Moreover, the application was designed with the user in mind and all ideas for improving the application for the user beyond a proof of concept are addressed in the future work section. Furthermore, the application was designed to support continued development. The code is annotated and compartmentalized for follow-on work to require little start-up effort, and lessons learned are provided in the appendix. The largest advantage of the developed application is that it exists today for examination and discussion. As satellite terminals and tablet applications continue to be developed this thesis will be available to provide a use case and example for direct imagery satellite tasking.

Following the discussion on design choices (Section B), the actual implementation of this proof of concept is presented in Section C. The development tools used are described in Section D and these can be used for future work.

B. PLATFORM SELECTION

The application developed in this thesis is simply one instantiation of how to allow warfighters or emergency personnel to directly request and task overhead assets. With that broad objective in mind, certain goals and requirements were developed to evaluate the design trades. Goals and requirements were identified by the developer after researching available imagery access and dissemination programs, current human-computer interaction ideologies, as well as available assets. The goals and requirements recognized by the developer and thesis advisors during early development are listed in Table 2. Each goal is addressed in the following trade discussions.

Goals
Portable hardware
User-friendly
Intuitive request process
Transmit request
Receive status
Receive information
Interact with image
Interact with raw image for analysis

Table 2. Goals identified for direct tasking of overhead assets.

Part of the desire to use portable hardware included the ability to use the device and application while moving or engaged in other activities. The chosen device should be able to be held in one hand and operated using the other hand. One should not expect that the warfighter or emergency personnel user would have a surface in the field to place the device. Whereas laptops typically require a flat surface to comfortably operate, tablets can be strapped to a forearm and operated by the opposite hand.

Additionally, the user should be able to send a request while holding another object. One can expect that the user may have other handheld equipment in support of their mission. For example the warfighter may have a handheld-night-vision scope, or other tactical gear. The user should not have to put down or away other equipment to use the application. The application should only require simple one-handed gestures, and the device should not require a stylus or pen. The experiments conducted in the Manual Multitasking Test for “ease of juggling” were considered when evaluating hardware (Oulasvirta and Bergstrom-Lehtovirta 2011). The Manual Multitasking Test asks a subject to perform a task on a device, for example, send a text message, while being constrained in one of twelve conditions emulating manual constraints frequent in everyday activities. Example manual constraints include using the non-preferred hand, holding another small object, not being able to support the device, or restricted movement of the shoulder, elbow and wrist. The experiments conducted by Oulasvirta and Bergstrom-Lehtovirta showed that use of a stylus decreased function while multitasking more than using a physical qwerty keyboard or touchpad qwerty keyboard. (Oulasvirta and Bergstrom-Lehtovirta 2011). Their findings support the use of a touchscreen device if the intent of the application is to be used by a warfighter or emergency personnel in the field. Either user would be expected to carry other items and could be required to use their arm or hand to perform a different task. The tablet and application was developed to be operated while propped on a surface, or held in one hand (see Figure 4). If the user is entering a request while holding the tablet in one hand some dexterity of the other hand is required.

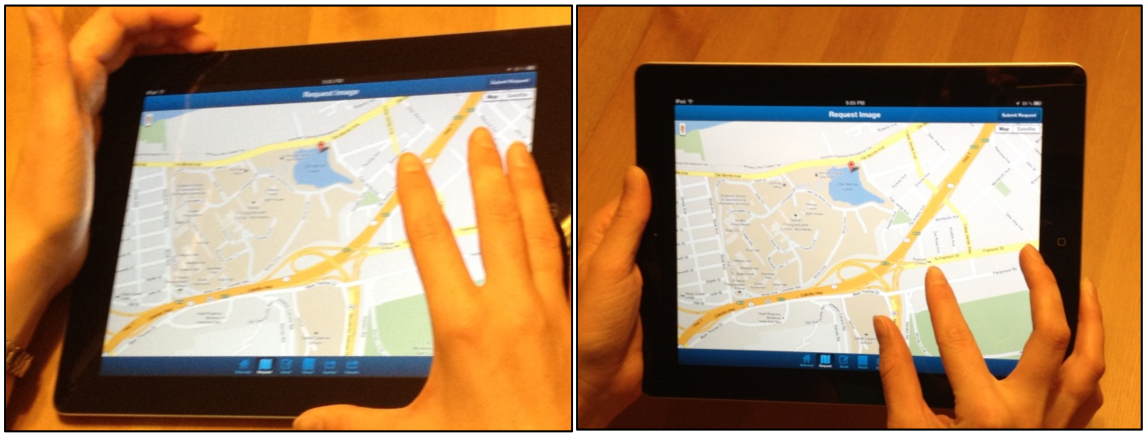


Figure 4. Tablet and application being used while propped on a surface or held in one hand and operated with the other.

Further research and human research approval will be necessary to fully evaluate the portability of the application developed here using the suggested Manual Multitasking Test or other industry standard assessments. However, the above cursory evaluation of the tablet and application’s portability suggest adequate convenience and the capability for the application to be made more functional if required by the target users.

The goal of user-friendliness and intuitive request process also played into the trade space of using a touchscreen. The subject of user-friendliness is related to the field of human computer interaction, or HCI. HCI includes, among other things, the study of modalities of direct manual input (for example touch and stylus), and indirect manual input (mouse and cursor). Tablets have typically used two types of direct manual input: touch and stylus entry. Touch and multi-touch entry has gained a lot of popularity, but what is an ideal input for some tasks is not necessarily ideal for others. For instance, using touch entry to turn the page of a book is more convenient than needing to pick up a stylus, but signing your name with your finger is not as precise as using a stylus. To truly be user-friendly, the input modality needs to be considered. There are drawbacks to using only touch input like the moderate precision as compared to using a pen, and the “Midas Touch Problem” (Hinckley, Pahud and Buston 2010). The “Midas Touch Problem” refers to the problems that highly responsive interfaces have in discerning deliberate movement from accidental movement. For a touch interface, common accidental movements are fingers brushing the screen, fingers on the screen to stabilize the device, or the finger can be left on the screen for too long (Hinckley, Pahud and Buston 2010). A stylus input may have a higher precision and fewer false positive inputs, but still has many drawbacks. The drawbacks include the stylus as a mechanical intermediary which needs time to be unsheathed and can be lost, as well as limited elementary inputs. The stylus can tap, drag, and draw a path, but multi-touch allows for other gestures like pinch and swivel (Hinckley, Pahud and Buston 2010). After a review of the two input modalities it was obvious that a touch input would be preferred to a stylus input for the direct satellite tasking application.

Several different application storyboards were considered before the final application layout was finalized. The end-goal was to decide which flow and layout provided the most intuitive request process. One possible flow is depicted in Figure 5. In the sample storyboard the user opens a welcome screen with four tabbed options along the bottom. The three second views, labeled "Page 1" "Page 2" and "Page 3," each correspond to a tab. The fourth tab on the welcome screen may be used to return back to the home screen, or for settings. The third views, labeled with the page number and lower case "a," could be a submit screen or verification screen. Page 1b, for example, would be a message to the user indicating a successful submission or similar. In this first draft of the application layout there was a desire to have separate accounts for each user. Once the account settings were entered, user information could be pre-populated into the request data. The status of the account, logged in or logged out, was to be indicated by the lock icon. As different application storyboards were considered, the more intuitive flows were selected.

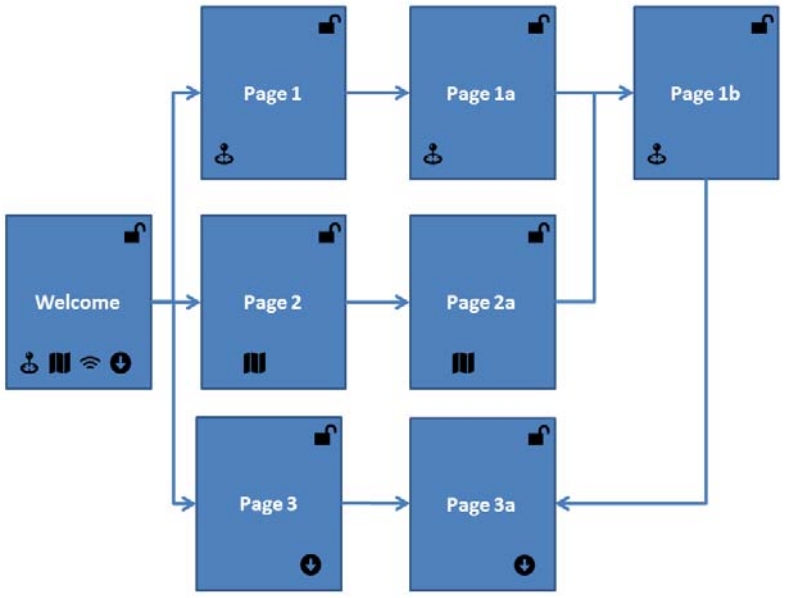


Figure 5. One possible application storyboard.

In the implementation section of this chapter the design concepts relating to the subjects above are addressed. For example, an email request form was created for user flexibility. It allows the user to enter the request via email instead of using the map request dialog. The final layout of the developed application can be seen in Figure 6.

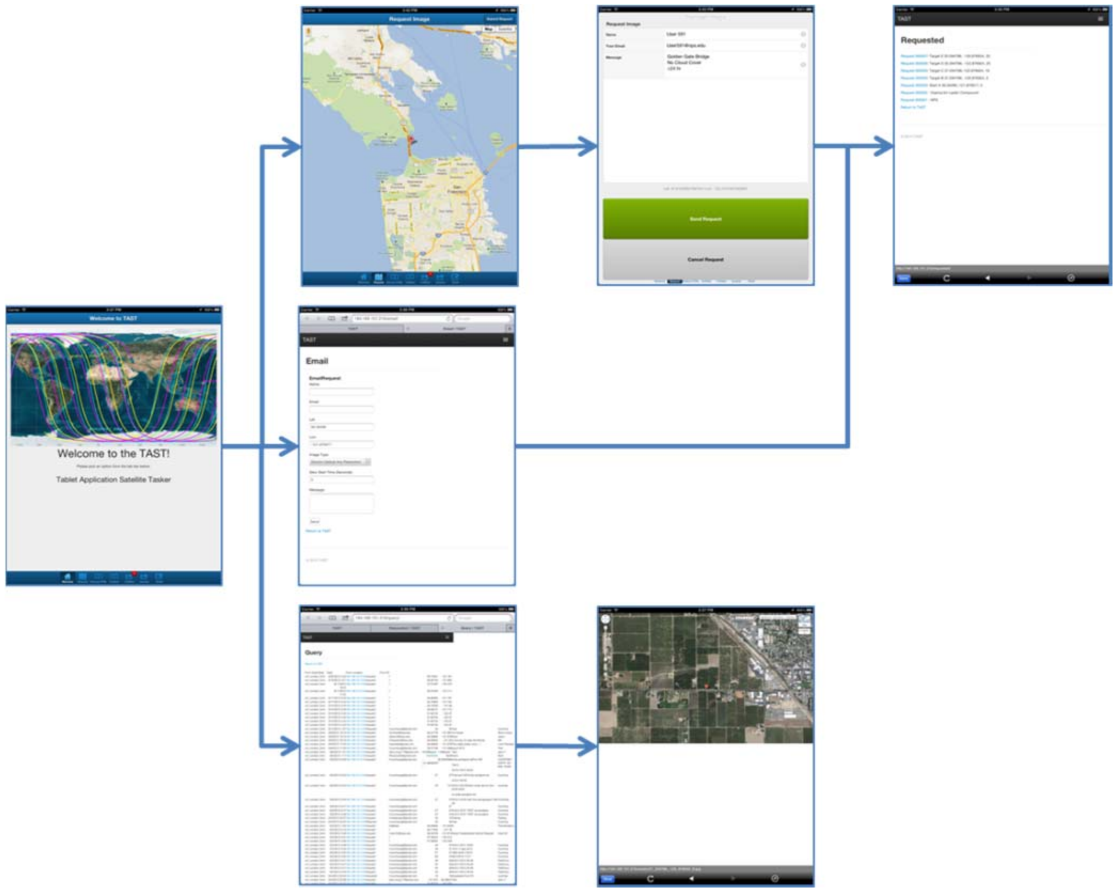


Figure 6. The final layout with application screen captures showing similarity to the draft storyboard of Figure 5.

The first page displays simple instructions and includes icons along the bottom linked to different pages. The first icon links to a request page that centers a map at the user’s current location. Once the user drops a pin on the map, the user can hit the submit request button and a form appears prepopulated with the latitude and longitude of the pin. The next page is a confirmation page listing the recent requests. In the middle row of the figure, the user can select an email request button (shown as Page 2 in Figure 5). The email request allows the user to enter a precise latitude and longitude. In the bottom row of the figure (shown as Page 3 and Page 3a in Figure 5), the user selects the fulfilled request icon. This icon brings up a page of recent requests with hyperlinks to the requested image if the satellite asset was able to fulfill the request. The user selects the hyperlink to the requested image and once downloaded, it can be manipulated using standard pinch and scroll controls. This layout of pages was contrived with consideration of the aforementioned goals. Beyond the application layout, the operating system, tablet hardware and software architecture were also selected with the goals considered.

1. Operating System

There are two primary mobile platforms available today, Apple's iOS and Google's Android. From the developer's point of view, there are two very different programming languages used on each. Apple's iOS uses Objective-C and Google's Android uses Java. Moreover, an application developed for one operating system cannot be used on the other. There was a choice between developing for a single platform, and developing native applications for both platforms. There was a third option to develop a mobile web application which would have minimized the amount of code required, but it would have also limited the features available for the application. Instead of trying to develop on multiple platforms, one platform had to be decided upon in the interest of time.

Several factors were evaluated for each platform; including the programming environment, development tools available, and hardware available. The programming environment offered by Apple's iOS Development Center is the Xcode package. Xcode includes an Interface Builder, iPad emulator, and development environment. Android offers the Android Development Tools plug-in for the Eclipse programming environment. It includes a UI design tool, GUI access to command line SDK tools, Java editor, and XML editor (Google 2012). These development tools greatly simplify the task of implementing a mobile application and focus on the individual developer with a goal to create an application quickly (Wasserman 2010). Objective-C is the high-level object orientated programming language used by Apple for the iOS operating system. Apple added many features to the Xcode Code Editor to make the work flow as user friendly as possible. The Code Editor uses code sense to quickly suggest the arguments for the function typed, and it has code folding to collapse code not currently being used. There are also quick help windows and documentation windows. The features of Apple's programming environment appeared to provide a better choice for application development.

Apple’s iOS platform was also chosen for the development tools available, including best practice guides, and frameworks. Apple’s Developer website includes an extensive “iOS Human Interface Guidelines” document in the library. This document includes everything from recommendations to only use approved gestures, to the detailed specifications required for icons. Apple’s documentation is aimed at providing guidelines so that the application will be user-friendly. The very existence of this document evidenced that there is a focus on helping developers create user-friendly applications. By having these guidelines available for iOS development, there was a set of defined “best practices” that could be used to develop the interface (Apple Inc. 2012). Future work for the tasking concept will require additional research into human computer interaction and its application to the device. However, the proof of concept development done here followed the usability guidelines to the extent possible while remaining within the scope of this thesis.

2. Tablet Hardware

There were no trades conducted in the selection of iOS devices as the only device available is Apple’s iPad. The tablet used for application provisioning was a 32 gigabyte Apple iPad 2. The installed software version was 5.1.1 (9B206).

The iPad tablet also had to be approved for provisioning. The first step is to have the team admin create an App ID and configure it with the SSL Certificate and Keys. Then the provisioning profile can be created by the team admin and installed on the developer’s machine. With the provisioning profile in place, the developer can request that a device be approved for provisioning. The unique device identifier, UDID, is entered into the provisioning profile by the team admin. The provisioning profile can now be downloaded by the developer and will allow certificate holders to test on the device (Apple 2012).

3. Software Architecture

Xcode offers templates for different types of iOS applications. The templates include the necessary code to build and run a basic application. The iOS templates include options that can create either an iPhone or iPad application, but require a lot of manual coding and effort to enable them operate on both platforms. The following sub-sections outline the available templates, and their relevance to the satellite tasking application.

a. Navigation-Based Template

The Navigation-based template sets up the basic framework to allow the developer to build menu-like navigation trees. This is done by combining a Navigation Controller object with table views. The cells of the table are selectable menu-like items. Navigation is handled by a controller that handles the trees and displays the title and relative back and forward buttons. The Navigation-based template is for iPhone only (Wentk 2011). An example application for this style template would be a list of authors for the user to select and the App would then display information about the author. This template creates basic menus that the user can select and retrieve information. The Navigation-based template was not chosen because it is only intended for use with static data.

b. OpenGL ES Template

The OpenGL ES template uses the OpenGL ES graphics subsystem and is used for complex custom user interfaces and games. It is intended for specialized high-performance graphics with complex 3D or 2D animations (Wentk 2011). This template was not selected for the satellite tasking application because there is not a requirement for animated graphics.

c. SplitView-Based Template

The SplitView-based template is the iPad equivalent of the Navigation based template for iPhone. The larger screen of the iPad allows for a larger list view of menu items. The split view displays differently in portrait orientation and landscape orientation. In the landscape orientation, the split view appears left. The portrait orientation the split view floats above the detail view (Wentk 2011). This template was not selected for the same reason the Navigation template was not used; it is best used for static data and the satellite tasking application needs to retrieve and display requested data.

d. Utility Application Template

The Utility application template is for the iPhone only and creates and information button at the bottom right of the screen. When the user taps the information button, a flip-side view appears with a navigation bar and a done button. There are two controllers, one for each view. The typical use for this template is for preferences and other application features. There is no equivalent template for the iPad (Wentk 2011). This template was not selected because it is iPhone-specific, and the simple navigation bar lacks a controller needed for the satellite tasking application.

e. View-based Application Template

The View-based application template can be used for the iPad or iPhone, but does not create a universal template for both applications. The developer must choose either iPad or iPhone. If iPhone is selected, the designed application will work on the iPad in an emulator mode that is in a half-size sub-window. The view-based application template is the most common template used by developers because code can be added immediately and it includes all the basic features for an app (a window, a view controller and a view (Wentk 2011). This template was not selected for the satellite tasking application because of its half-size appearance on the iPad.

f. Window-Based Application Template

The Window-based application template includes only window with a single label. This iOS template can create a single application that runs on both the iPad and iPhone platform. The template actually creates two separate applications, with separate nibs, and application delegates. The application loader on the iPad or IPhone selects the appropriate nib to run on start-up of the application (Wentk 2011). This template was not selected for the satellite tasking application because it did not include a view controller required for the application features.

g. Tab Bar Application Template

The Tab Bar application template can be used on the iPhone or iPad. The user taps buttons on the bottom tab bar to select different views. The template includes two view controllers and more can be added by the developer (Wentk 2011). Of the available templates, a modified version of the tab bar application template was used. The tab bar application allows the user to tap buttons on the bottom tab bar to select different views. This template was selected so the user could reach nearly all pages from the introduction page; this simplifies hierarchies and allows the user to complete their task quickly.

C. IMPLEMENTATION

This section describes the look and feel of the developed application and explains the user interface. The specific code tools used are described later in Section D.

1. First Impression

On the iPad the user opens the application named "TAST." TAST is almost a homophone for the word "task" and is the acronym for "Tablet Application Satellite Tasker." When the application first launches, the launch image is displayed, followed by the initial loading indicator that signifies the application is loading. This is useful feedback to the developer that scripts are loading correctly, and useful to the user as positive feedback that the program is actively launching and not frozen. The screenshot from the application simulator can be seen in Figure 7. The application is configured for portrait and landscape uses, but figures of the landscape use are not provided because they offer no additional information.

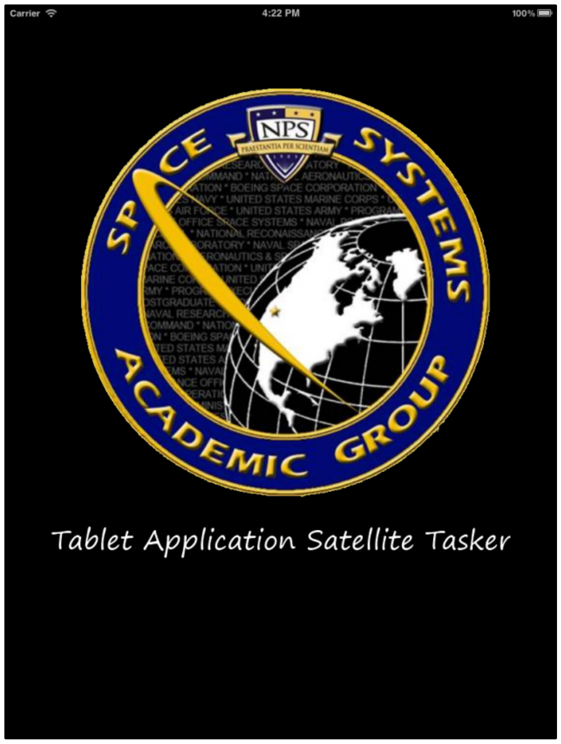
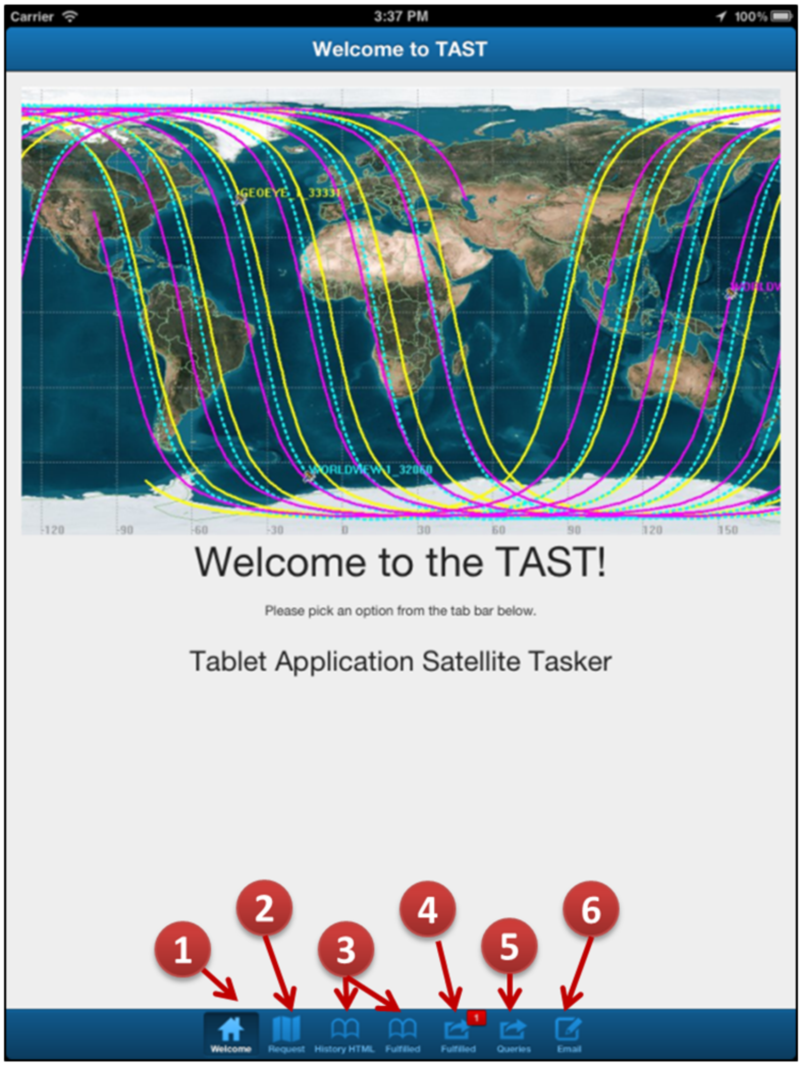

Carrier
4:22 PM
100%
SPACE SYSTEMS
ACADEMIC GROUP
NPS
PRAESTANTIA PER SCIENTIAM
Tablet Application Satellite Tasker

Figure 7. Screenshot of launch icon of the developed TAST application.

Once the application loads, the home-page is displayed. The home-page includes a place-holder graphic created using Satellite Took Kit, a software product by Analytical Graphics, Inc. The graphic is the ground trace of GeoEye-1, Worldview-1, and Worldview-2 on the 2D map viewer. This graphic has no utility in this iteration of the TAST application. However, it is included because future work should insert the ability to show the ground tracks of available imaging assets. The ground tracks should be animated and show the swath of available targets. This visual tool would help the trained user assess the likelihood of being able to request an image collection. Below the graphic and title there are instructions to the user and several tabs for the user to select. The homepage view can be seen in Figure 8.



Welcome to TAST

Welcome to the TAST!

Please pick an option from the tab bar below.

Tablet Application Satellite Tasker

Figure 8. Screenshot of TAST homepage with seven tab-bar buttons.

The tab bar navigational model was selected because of its popular use in iPhone and iPad applications. The iOS Human Interface Guidelines suggest using a toolbar for controls that perform actions related to the objects in the screen or view. The first button in the tool bar, marked one in Figure 8, returns the user to the home screen shown. The second button opens the map for the user to select a location. The buttons marked with a three open a page in the application with sample pictures from the server. Button four opens the Safari application to display sample pictures, and button five also opens Safari to a webpage showing the requests on the server. Button six opens the email request form. The rule of thumb in application design is no more than five icons on the tab bar.

Function over form was, however, chosen for this application start-up page. It seemed more important to show the possible implementations in this proof of concept than to strictly follow the human interface guidelines. Partly because, the rule stems from the available space on the iPhone, but since this is an iPad application, the extra buttons still fit on the screen. Additionally, users can see all the available tabs to select from and avoids the problem of users not finding extra features is avoided. Future work should focus on HCI aspects and include user testing to garner feedback on icon recognition, placement, and function.

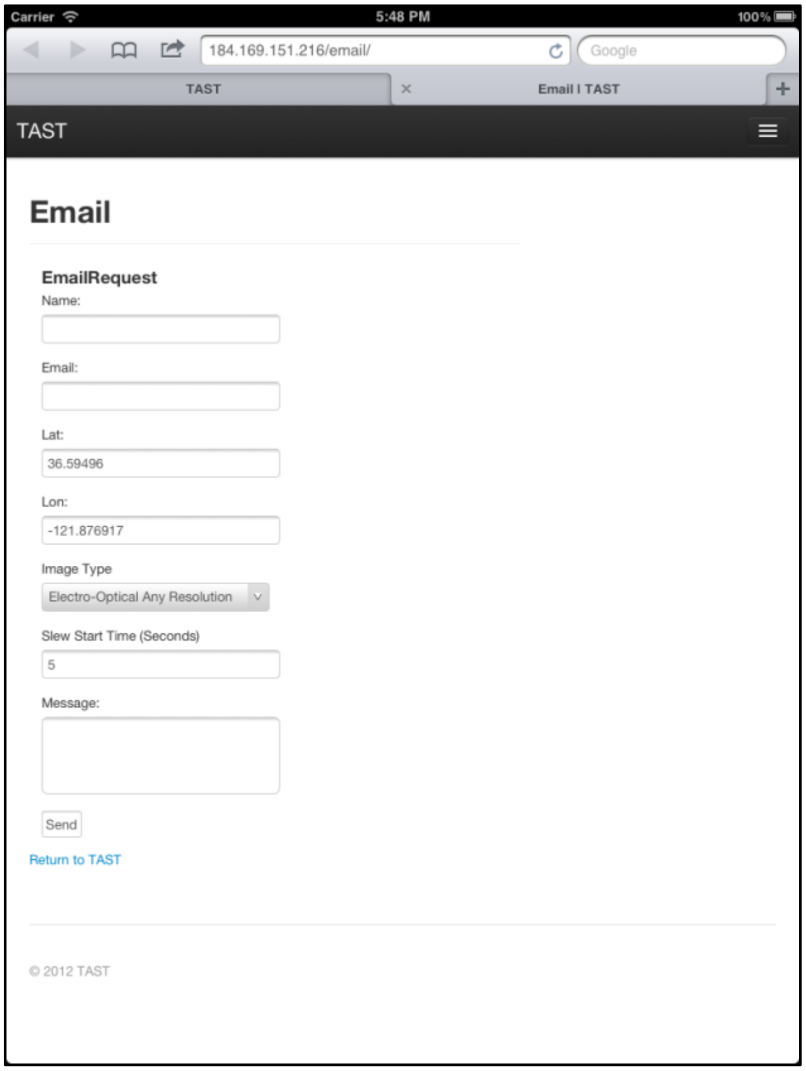
The following section elaborates on the options the user has to submit a request to the satellite. The user can request a particular location by entering coordinates, or by selecting a location on a map. These functions are all available from the tab-bar buttons on the homepage.

2. Request Structure

a. Email Request

An email request form was created so that the user had the flexibility to enter any coordinates without having to find the location on the map. This feature could also be useful if limited bandwidth is available and the map was not previously loaded onto the device. Having said that, a released version of the tasking software should allow relevant maps and graphics to be pre-loaded by the user.

For the user to submit an email request, they tap on the compose icon with title "Email" under it, number six in Figure 8. The form opens for users to input their information, requested latitudes and longitudes and desired parameters for the image. The form is shown in Figure 9.


Carrier 5:48 PM 100%

184.169.151.216/email/

TAST | Email | TAST

TAST

Email

EmailRequest

Name:

Email:

Lat:

36.59496

Lon:

-121.876917

Image Type

Electro-Optical Any Resolution

Slew Start Time (Seconds)

5

Message:

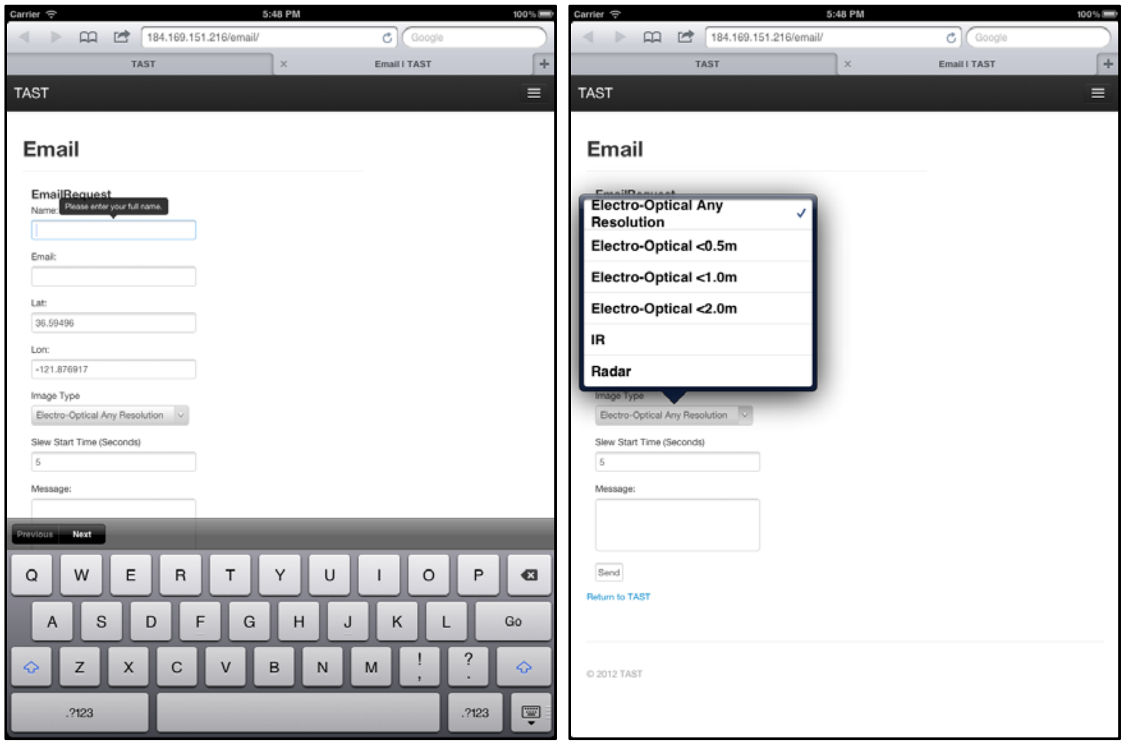
Send

Return to TAST

© 2012 TAST

Figure 9. Screenshot of TAST email request form.

The form opens in Safari. The external form, vice a native form, was used for several reasons. One reason is that the form contents could be easily saved on the server. Setting up the form externally on the server allowed easy flexibility of the form contents. As the development of the application progressed and more research on the topic was conducted, the desired information from the user changed. The current form asks for the user's name, email, latitude and longitude of request, the image type/resolution requirements, and slew start time. There is an additional message field for any special instructions or to add a name for request. Details of the user interface while entering data into the input forms can be seen in Figure 10. If the user taps on an input field, they are given instructions in a pop-up field. The pop-up field is provided for clarification, and so that description by way of an additional user's manual would not be necessary. When the user taps on a pull-down menu, the menu pops-up and allows the user to select what type of image they are requesting and relative ground sample distance, GSD.



Name:

Email:

Lat: 36.59496

Lon: -121.876917

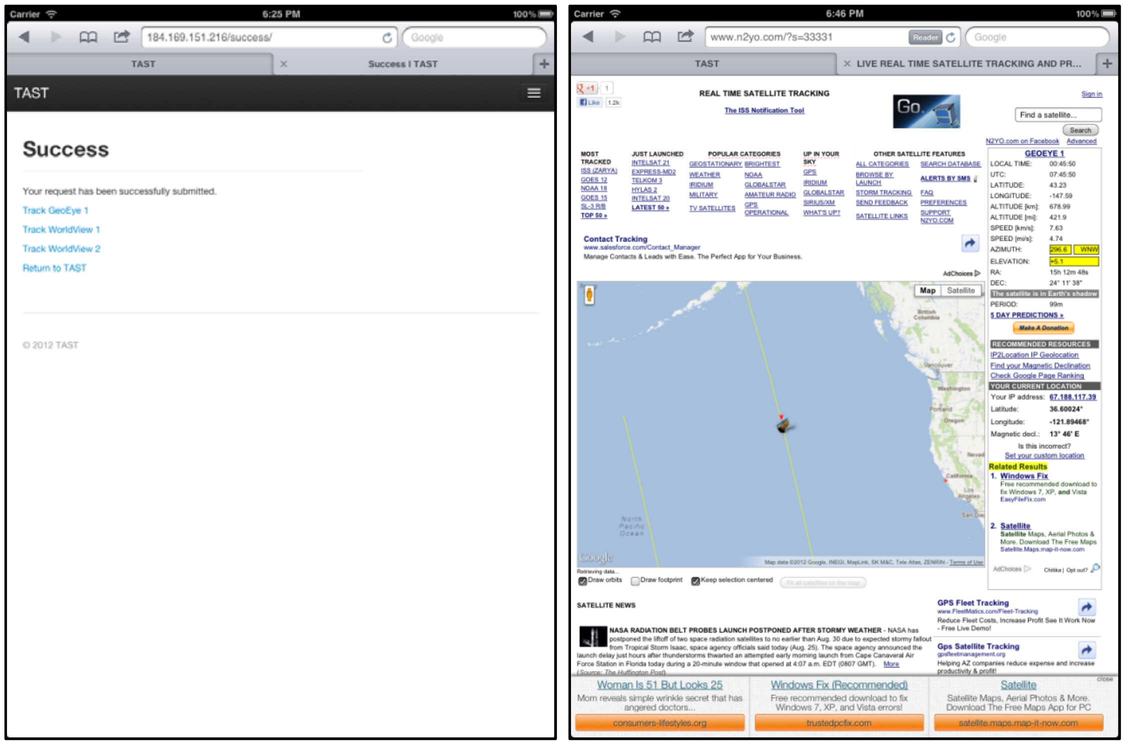
Image Type: Electro-Optical Any Resolution

Slew Start Time (Seconds): 5

Message:

Figure 10. Screenshots of the TAST email request form field details and keyboard activation.

The current settings on the form input fields do not require any of the fields to be filled out before submission, and there are only exaggerated limits placed on the maximum number of characters allowed in each field. Once the user has completed the form to their satisfaction, they tap the submit button and are taken to another page outside of the TAST application. The successful submission page includes links to online satellite trackers and a link to re-launch the application on the iPad. A future version of the application should integrate this step as part of the application package to avoid having to switch between applications. The successful submission page and satellite tracking page of GeoEye 1 are shown in Figure 11. These links are included in the application because a satellite tracker would be useful if incorporated into the final application. With the satellite tracker, a trained user would be able to see the available satellite location and estimate when a pass of the desired location could occur. Ideally, the application could also automatically calculate the pass times for the user.



Success

Your request has been successfully submitted.

Track GeoEye 1
Track WorldView 1
Track WorldView 2
Return to TAST

© 2012 TAST

Contact Tracking
www.salesforce.com/Contact_Manager
Manage Contacts & Leads with Ease. The Perfect App for Your Business.

GEOEYE 1

LOCAL TIME:	00:45:50
UTC:	07:45:50
LATITUDE:	43.23
LONGITUDE:	-147.59
ALTITUDE [km]:	678.99
ALTITUDE [mi]:	421.9
SPEED [km/s]:	7.63
SPEED [mi/s]:	4.74
AZIMUTH:	296.6 WNW
ELEVATION:	+5.1
RA:	15h 12m 48s
DEC:	24° 11' 38"
The satellite is in Earth's shadow	
PERIOD:	99m

YOUR CURRENT LOCATION

Your IP address:	67.188.117.39
Latitude:	36.60024°
Longitude:	-121.89468°
Magnetic decl.:	13° 46' E

Is this incorrect?
Set your custom location

Related Results
1. Windows Fix
Free recommended download to fix Windows 7, XP, and Vista
EasyFileFix.com

2. Satellite
Satellite Maps, Aerial Photos & More. Download The Free Maps
Satellite.Maps.map-it-now.com

AdChoices Chitika | Opt out?

SATELLITE NEWS

NASA RADIATION BELT PROBES LAUNCH POSTPONED AFTER STORMY WEATHER - NASA has postponed the liftoff of two space radiation satellites to no earlier than Aug. 30 due to expected stormy fallout from Tropical Storm Isaac, space agency officials said today (Aug. 25). The space agency announced the launch delay just hours after thunderstorms thwarted an attempted early morning launch from Cape Canaveral Air Force Station in Florida today during a 20-minute window that opened at 4:07 a.m. EDT (0807 GMT). More
(Source: The Huffington Post)

GPS Fleet Tracking
www.FleetMatics.com/Fleet-Tracking
Reduce Fleet Costs, Increase Profit See It Work Now - Free Live Demo!

Gps Satellite Tracking
gpsfleetmanagement.org
Helping AZ companies reduce expense and increase productivity & profit!

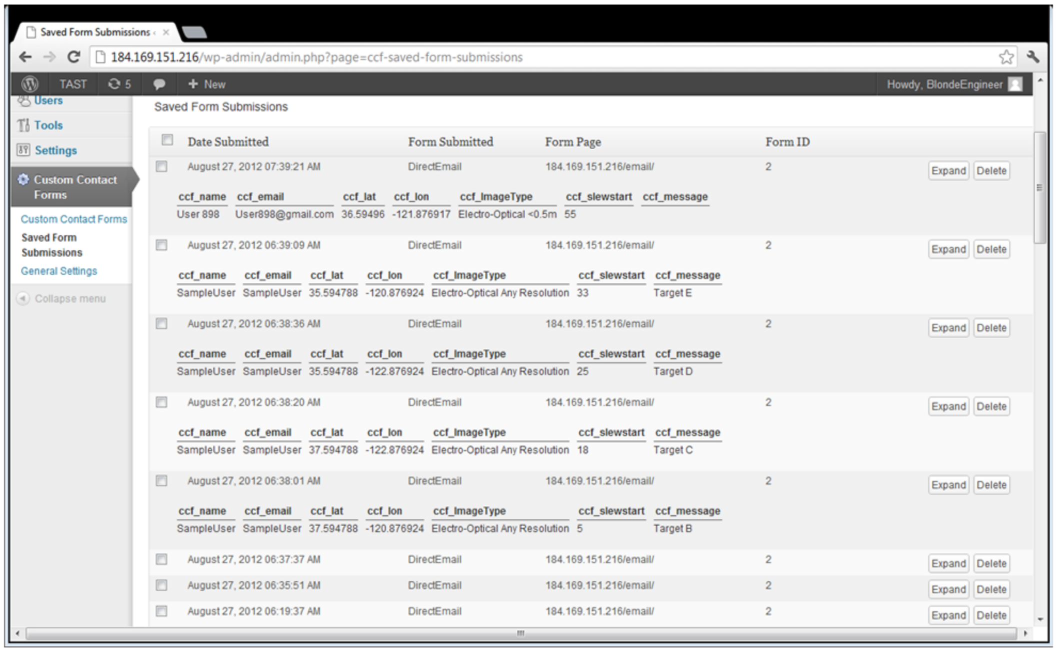
Woman Is 51 But Looks 25
Mom reveals simple wrinkle secret that has angered doctors...
consumers-lifestyles.org

Windows Fix (Recommended)
Free recommended download to fix Windows 7, XP, and Vista errors!
trustedpcfix.com

Satellite
Satellite Maps, Aerial Photos & More. Download The Free Maps App for PC
satellite.maps.map-it-now.com

Figure 11. Screenshots of the TAST email request success page and tracking page of GeoEye 1.

Successful requests are emailed to the specified address on the server and are also stored on the server. A sample of the form submissions appearance is exhibited in Figure 12. The form submissions can also be downloaded in a comma-separated values file format. The downloaded data can be seen in Table 3. The csv-file format is in plain text and can be used by many applications. Most applications can support a csv-file import and this could be useful for future implementation of the application data.



Saved Form Submissions

Date Submitted	Form Submitted	Form Page	Form ID
August 27, 2012 07:39:21 AM	DirectEmail	184.169.151.216/email/	2

ccf_name	ccf_email	ccf_lat	ccf_lon	ccf_ImageType	ccf_slewstart	ccf_message
User 898	User898@gmail.com	36.59496	-121.876917	Electro-Optical <0.5m	55	

Date Submitted	Form Submitted	Form Page	Form ID
August 27, 2012 06:39:09 AM	DirectEmail	184.169.151.216/email/	2

ccf_name	ccf_email	ccf_lat	ccf_lon	ccf_ImageType	ccf_slewstart	ccf_message
SampleUser	SampleUser	35.594788	-120.876924	Electro-Optical Any Resolution	33	Target E

Date Submitted	Form Submitted	Form Page	Form ID
August 27, 2012 06:38:36 AM	DirectEmail	184.169.151.216/email/	2

ccf_name	ccf_email	ccf_lat	ccf_lon	ccf_ImageType	ccf_slewstart	ccf_message
SampleUser	SampleUser	35.594788	-122.876924	Electro-Optical Any Resolution	25	Target D

Date Submitted	Form Submitted	Form Page	Form ID
August 27, 2012 06:38:20 AM	DirectEmail	184.169.151.216/email/	2

ccf_name	ccf_email	ccf_lat	ccf_lon	ccf_ImageType	ccf_slewstart	ccf_message
SampleUser	SampleUser	37.594788	-122.876924	Electro-Optical Any Resolution	18	Target C

Date Submitted	Form Submitted	Form Page	Form ID
August 27, 2012 06:38:01 AM	DirectEmail	184.169.151.216/email/	2

ccf_name	ccf_email	ccf_lat	ccf_lon	ccf_ImageType	ccf_slewstart	ccf_message
SampleUser	SampleUser	37.594788	-120.876924	Electro-Optical Any Resolution	5	Target B

Date Submitted	Form Submitted	Form Page	Form ID
August 27, 2012 06:37:37 AM	DirectEmail	184.169.151.216/email/	2
August 27, 2012 06:35:51 AM	DirectEmail	184.169.151.216/email/	2
August 27, 2012 06:19:37 AM	DirectEmail	184.169.151.216/email/	2

Figure 12. Saved form submissions on tasking server.

Form Submitted	Date	Image Type	Email Address	Lat (degrees)	Long (degrees)	Request Name	User Name	Slew Start Time (sec)
DirectEmail	8/27/2012 7:39	Electro-Optical <0.5m	User898@gmail.com	36.59496	-121.877		User 898	55
DirectEmail	8/27/2012 6:39	Electro-Optical Any Resolution	SampleUser	35.59479	-120.877	Target E	SampleUser	33
DirectEmail	8/27/2012 6:38	Electro-Optical Any Resolution	SampleUser	35.59479	-122.877	Target D	SampleUser	25
DirectEmail	8/27/2012 6:38	Electro-Optical Any Resolution	SampleUser	37.59479	-122.877	Target C	SampleUser	18
DirectEmail	8/27/2012 6:38	Electro-Optical Any Resolution	SampleUser	37.59479	-120.877	Target B	SampleUser	5
DirectEmail	8/27/2012 6:37	Electro-Optical Any Resolution	SampleUser	36.59496	-121.877	NA	SampleUser	0
DirectEmail	8/27/2012 6:35	Electro-Optical <1.0m	User592@nps.edu	37	121	NA	User 592	5
DirectEmail	8/27/2012 6:19	Electro-Optical <1.0m	User591@nps.edu	37.12121	112.1212		User 591	5

Table 3. Downloaded form submissions from server.

The request data is available for the satellite in two forms, email and data on the server. Operationally, the emailed form data could be sent anywhere, and parsed out to generate satellite commands. In this proof of concept the data is downloaded from the server, and a simple macro is used to parse the data into the text-files required for the MATLAB spacecraft simulation. Operationally, the macro would be replaced by a scheduler instead of MATLAB loading the data from the text-files. The scheduler would push the relevant data to the command buffer in any one of the participating imaging assets. Before implementation with real satellites, a security feature would have to be added to check that the request was received correctly and that the user is authorized to request the image. This and a plethora of other compatibility and security issues would need to be resolved prior to operationally using the table tasking concept.

b. Map Request

A map request feature was created so that the user has the flexibility to tap on any location on the map without knowledge of the coordinates. This feature also allows the user to see existing satellite imagery. If the user does not have a need for more current data, the existing map may be sufficient to allow for mission execution.

To use the map feature, the user taps on the maps icon with title “Request” under it, see number 3 in Figure 13. The Google Map opens within the application. Above the map is a title bar that says “Request Image,” and has a button “Submit Request.” The “Submit Request” button is disabled until a marker is placed on the map. A marker is placed on the map when the user taps on a location. An example marker is labeled with the number 2 in Figure 13.   Once the marker position is saved to a variable, the “Submit Request” button is activated. The “Submit Request” button is labeled 1 in Figure 13.

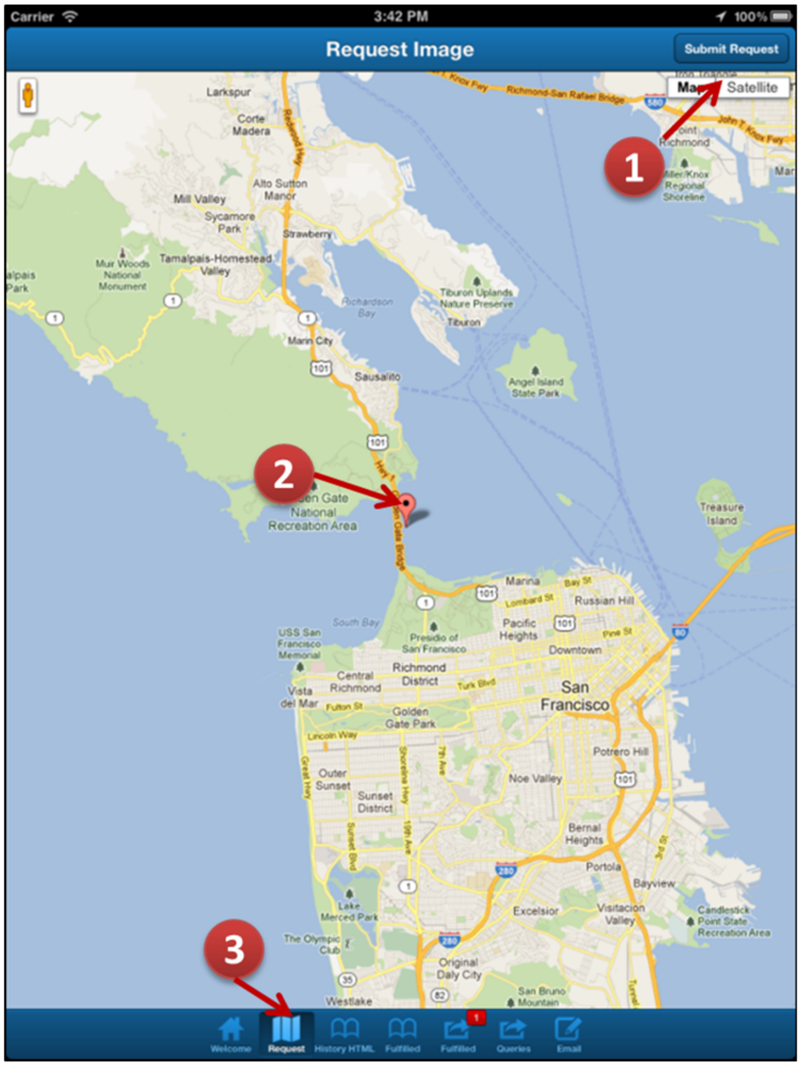
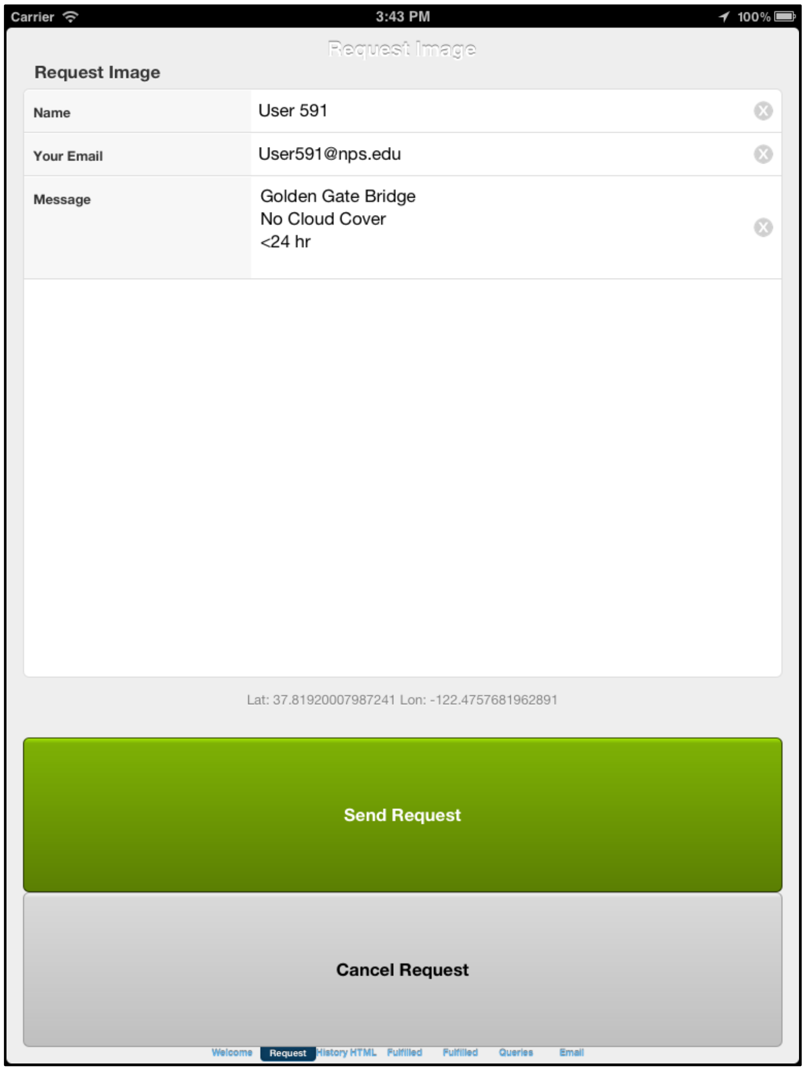


Figure 13. Screenshot of the TAST map request page with a placed marker (2) and activated navigation bar button (1).

When the “Submit Request” button is tapped, the coordinates are pre-loaded into the request form and the user is asked for additional information. The user is asked for their name, email, and is given a space to enter any additional parameters. Future development of this application should collect feedback from users and imaging asset operators on what parameters should be selectable in the form. In addition an interface that allows a request to be populated and sent without the need to interact with a form may be desirable for the user.


Carrier 3:43 PM 100%

Request Image

Request Image

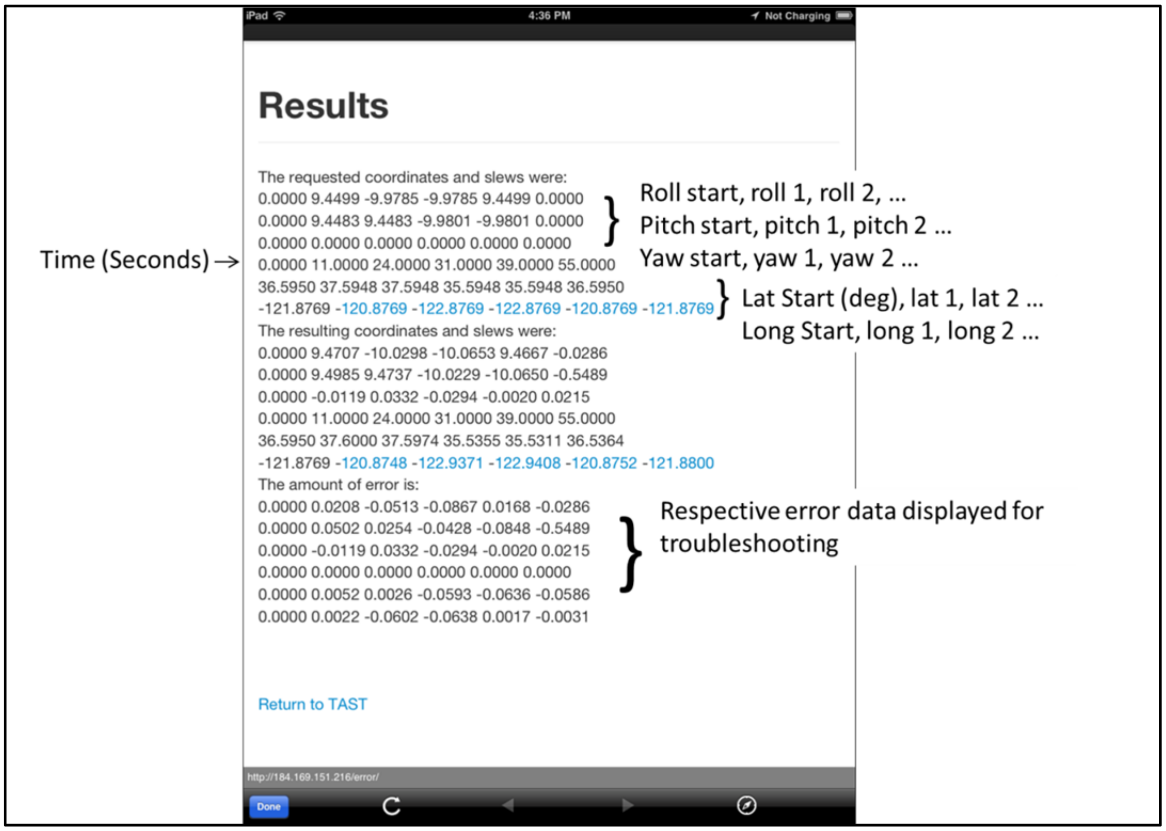
Name	User 591
Your Email	User591@nps.edu
Message	Golden Gate Bridge No Cloud Cover <24 hr

Lat: 37.81920007987241 Lon: -122.4757681962891

Figure 14. Screenshot of the TAST request form after a point is selected on the map.

3. Receiving Data

Once the request has been processed, the satellite tasked, and an image is made available, the data is uploaded to the server and the TAST application can retrieve the data. The user can tap the “Results” tab and the server is pinged for an updated “results.txt” file. The text file is loaded into a page that the application can display using hypertext preprocessor scripting. The screen is loaded with the requested coordinates and slews, the resulting coordinates and slews, and the amount of attitude error during imaging. The screen can be seen in Figure 15. The display of the results is a demonstration of data being passed back to the application. The data could be anything, an image, or a message about the status of the request. The most useful information for the tactical user would be the image file, along with any analysis if available.



```
Results

The requested coordinates and slews were:
0.0000 9.4499 -9.9785 -9.9785 9.4499 0.0000
0.0000 9.4483 9.4483 -9.9801 -9.9801 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 11.0000 24.0000 31.0000 39.0000 55.0000
36.5950 37.5948 37.5948 35.5948 35.5948 36.5950
-121.8769 -120.8769 -122.8769 -122.8769 -120.8769 -121.8769
The resulting coordinates and slews were:
0.0000 9.4707 -10.0298 -10.0653 9.4667 -0.0286
0.0000 9.4985 9.4737 -10.0229 -10.0650 -0.5489
0.0000 -0.0119 0.0332 -0.0294 -0.0020 0.0215
0.0000 11.0000 24.0000 31.0000 39.0000 55.0000
36.5950 37.6000 37.5974 35.5355 35.5311 36.5364
-121.8769 -120.8748 -122.9371 -122.9408 -120.8752 -121.8800
The amount of error is:
0.0000 0.0208 -0.0513 -0.0867 0.0168 -0.0286
0.0000 0.0502 0.0254 -0.0428 -0.0848 -0.5489
0.0000 -0.0119 0.0332 -0.0294 -0.0020 0.0215
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0052 0.0026 -0.0593 -0.0636 -0.0586
0.0000 0.0022 -0.0602 -0.0638 0.0017 -0.0031

Return to TAST
```

iPad
4:36 PM
Not Charging
Time (Seconds)→
Roll start, roll 1, roll 2, ...
Pitch start, pitch 1, pitch 2 ...
Yaw start, yaw 1, yaw 2 ...
Lat Start (deg), lat 1, lat 2 ...
Long Start, long 1, long 2 ...
Respective error data displayed for troubleshooting
http://184.169.151.216/error/
Done

Figure 15. Screenshot of TAST results display.

D. DEVELOPMENT TOOLS

In this section, the software tools used to build the TAST application are described. A number of modules beyond beyond those native to the Xcode suite had to be used to achieve the desired functionality.

1. Sencha Touch 2

Sencha Touch 2 is a developer tool for mobile web applications. Specifically it is an HTML5 mobile application framework. The tool automates a large part of the application generation and application building process. The files that are included as part of the free download include a getting started example application. The getting started application includes a professional looking loading indicator and a basic example of a main page JavaScript code. The simple application includes a home page, a contact form and a simple list to fetch recent blog posts (Sencha Inc. 2012). This example application is the basis for the tablet application created for this proof of concept.

This tool was selected for its ability to meet several of the goals identified. It aided in creating a user-friendly application, while also being able to submit data through forms-based infrastructure.

Submitting data through forms partially meets the requirement for transmitting requests. Form input was chosen for this proof of concept because of its popularity in systems and applications (Molich and Nielsen 1990). The popularity of forms made their use rational in two ways. For one, the data transmitted can be easily parsed and then submitted to existing platforms in any format required. Secondly, it can be assumed that most tablet users are familiar with forms. In future iterations of the form, it can be made more error-tolerant and provide more carefully phrased informative messages to the user.

The user requests do not necessarily need to be transmitted through forms. Other approaches are possible. One alternative approach that was considered was to allow the user to send a text message with their request. The identified downfalls of allowing a free-form text message request include incomplete parameters, missing parameters and more complex algorithms need to parse the information at the receiving side. With forms, the user receives instantaneous feedback that their request may be incomplete. The instantaneous feedback results in fewer incomplete requests and retransmissions of requests. Additionally, the platform receiving the requests needs to be smart enough to parse data that is not in a uniform format. Form submittal was chosen for the above reasons; better feedback to the user and easier algorithms for receiving platforms.

There are many other development tools available but Sencha was selected based on information available at the time and seemed the best fit for requirement fulfillment. However, using the Sencha tool added a layer of abstraction to application coding. This abstraction created hurdles in debugging and quite possibly made the coding more complex than necessary.

2. Apache Cordova (formerly PhoneGap) and Plugins

Apple allows development of applications for their iOS in the programming language of Objective-C. Typically Apple users, and developers alike, saw learning Objective C as barrier for many to begin developing. Therefore, a group of attendees at iPhoneDevCamp (a conference started in 2007 for Apple developers) decided to create a framework for web developers to easily create applications for iOS devices (Frakes 2009). One of these frameworks was PhoneGap originally created by Rob Ellis and Andre Charland (LeRoux 2012). Their idea was to embed a webkit into a native Objective-C application and build a JavaScript API that calls the native iPhone functions such as geolocation and accelerometer. PhoneGap expanded to include more of the native iPhone functions and eventually the PhoneGap codebase was donated to Apache Software Foundation (LeRoux 2012).

In addition to making use of the Cordova development tools, an additional plugin to the suite called "Childbrowser" was utilized. The requested images are loaded into a database and uploaded to a server. A simple webpage was created to display the resulting images, and the intent was to use the PhoneGap plugin "ChildBrowser" open the webpage within the application. It was confirmed that the .m, .h and .bundle files were all moved to the Plugins folder of the project. The correct key/value pairs were listed in the Cordova.plist file, and the server was also added to the external host inputs. However, there was a deprecation notice in the Cordova JavaScript file that said the plugins will be removed in the subsequent version. Therefore, the sample webpage could not be loaded by the TAST application. The assumption is that during the various iOS updates, PhoneGap/Cordova updates, and ChildBrowswer plugin updates, one version became incompatible with the other. This is a lesson learned for future application development. Due to the open-source nature of PhoneGap/Cordova and other developer tools, updates come out often, version control becomes cumbersome, and documentation of changes may not exist. Debugging version differences can become very time consuming.

Using the Cordova Child Browser plug in is obviously not the only way to send information to a webserver. There are server proxies that can handle the marshaling of data to a server, and there are client proxies that use memory to store data in the browser's memory or local storage when available. During the development of the TAST application many different attempts were made to send the data through proxies. Roadblocks relating to server permissions were encountered as well as difficulty in coding the scripts necessary to store and retrieve information. The final solution for this proof of concept used both a form submit function and the native web browser, Safari, on the iPad and to send the information, a non-ideal solution at best. The difficulties encountered with the sample task of integrating a data pipeline into the TAST application is illustrative of the challenges many developers face in bringing a new concept to fruition.

3. Google API

One feature of the tablet application is the ability to load a map of the current position and select a point on the map to request an image of that point. The Google Map API V3 was integrated into the tablet application to add this feature. A listener function is loaded to catch the touch event on the tablet. The touch event triggers a function that places a marker at the touched latitude and longitude. The latitude and longitude of the marker are saved to a variable (e.latLng) and loaded into the request form. For the map to load and the functions to work the Google URLs also need to be listed in the Cordova.plist file in the Resources file. The URLs are added as strings under the "ExternalHosts" array (Engvall 2012). Once the URLs are added, the scripts are loaded and the user allows the application to use the current location, the map will load on the iPad.

A future work suggestion related to the Google Map API is to add a geocoding feature. Geocoding is the process of turning an address into a geographic point. Adding this feature to the application would allow the user to type in an address and receive the coordinates. Geocoding is supported in the Google Maps API, but was not integrated into the application.

4. Server

The coordinates and form data captured by the application are sent to an Amazon EC3 Server meant to represent the satellite tasking server. The EC3 server is a free server offered Amazon Web Services. Installed on the server is an open source content management system. A content management system was necessary to store the requested data in a centralized location. Other approaches are possible but not explored. The content management system used is WordPress. The WordPress plugins made it possible to submit data to the server and store data on the server. An additional SMTP server is used to relay the data from the application to an email address. The SMTP relay services used was SendGrid.

E. SUMMARY

This chapter explained the background for the engineering decisions made to develop the tablet tasking application. The designed application is for an iPad using a tab-based template from Sencha-Touch. Design decisions were made to make the application user-friendly and some research was done to support these decisions. For the actual implementation more research and input from end-users, the warfighters and emergency responders, is recommended. The next chapter explains the satellite model that is acting as the satellite being tasked by the TAST application, followed by a chapter that shows the integration of the model and the TAST application.

THIS PAGE INTENTIONALLY LEFT BLANK

III. SPACECRAFT MODEL

A. OVERVIEW

A simulation of a CMG spacecraft is used to validate the image collection segment of the tasking model. It is an important step in proving the concept before it is tested on a real ground test bed. The outputs and signals of the simulation should be reviewed to gain confidence that the tasking model will work and will not damage the real hardware. The simulation trajectory for slewing must be validated to ensure that real hardware can perform as directed. Other internal values, like torques and angular momentum should also be observed to validate the model itself. Once the spacecraft simulation has been fully vetted, the full scope of the tasking model can be tested on test bed hardware.

The spacecraft model described in this chapter is split up into subsystems and is described logically from the command processing through the torque command generation to the control moment gyros, followed by the dynamics and kinematics. The feedback control and disturbances are described afterward with an explanation of where they are inserted into the model.

B. COMMAND PROCESSING

A simple MATLAB script retrieves the text files containing the coordinates and timestamps of the image requests and converts them into slew commands based on the location of the satellite. The MATLAB script first retrieves the location of the satellite from a text file containing the latitude and longitude of the satellites nadir pointing vector. The MATLAB code is available in the Appendix. The satellite's altitude is pre-programed into the SIMULINK model in the InitFcn Callbacks. The altitude variable, h_{orb}, is defined as 681,000 meters, simulating the commercial vehicle GeoEye-1 (Miller 2011).

The slew-commands assume a flat earth model, shown in Figure 16. For small slews (of a low orbiting satellite) from nadir the flat earth approximation is a sufficiently good estimate for pointing for this application (Zipfel 2000, 368-370). The error in slew angle and centered coordinates on the ground would be absorbed by the overall swath of the image detector.

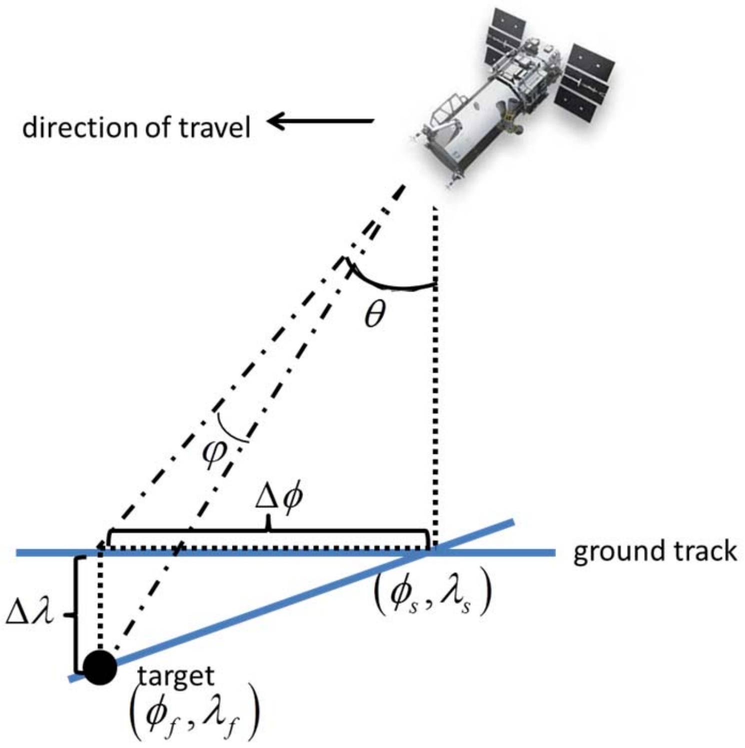


Figure 16. Flat earth model and slew angle determination.

With the sub-satellite location (ϕ_s, λ_s), and satellite heading (direction of travel), the position of the target on a planar Earth is computed using right triangles (Hodgson and Kar 2008). The equations to approximate the slew angle are shown in Equations (1)–(4).

$$\Delta\phi = (\phi_f - \phi_s) * \frac{\pi}{180} \tag{1}$$

$$\Delta\lambda = (\lambda_f - \lambda_s) * \frac{\pi}{180} \tag{2}$$

$$\theta = \tan^{-1}\left(\frac{\Delta\lambda}{h_{orb}}\right) \tag{3}$$

$$\varphi = \tan^{-1}\left(\frac{\Delta\phi}{h_{orb}}\right) \tag{4}$$

The distance from the sub satellite point to the target location can be calculated by estimating the distance between the two latitudes and longitudes. The initial latitude, ϕ_s, is subtracted from the final latitude ϕ_f, and converted into radians. The distance, for the longitude is calculated similarly. The necessary angle of roll, θ, is calculated by taking the inverse tangent of the change in longitude divided by the altitude of the satellite, h_{orb}. Then, the necessary angle of pitch, φ, is calculated by taking the inverse tangent of the change in latitude divided by the altitude of the satellite, h_{orb}. This flat-earth slew model introduces some error. The error was analyzed and the model was deemed sufficient for illustrative purposes of the satellite model.

C. TRAJECTORY GENERATION

After the roll and pitch angles are determined for the requested coordinates, the slew trajectory is generated. There is a wait time before the maneuver is executed to observe that the spacecraft model is initially quiescent. The trajectory vector $\theta\ (t)$ is actually made up of three different equations for each maneuver. Each maneuver is six seconds long and each equation is active for two seconds. A six second maneuver was chosen based on a maximum 18° slew, which covers approximately two degrees in longitude. The 18° slew would require a slew rate of three degrees per second. DigitalGlobe advertises that WorldView-2 can slew at a rate of 3.5 degrees per second (DigitalGlobe 2015). Therefore, the slew rate is not beyond industry standards and is reasonable for this proof of concept, but it is significantly faster than the capability of GeoEye-1. DigitalGlobe advertises that GeoEye-1 can slew 200 km on the ground in about 20 seconds, versus WorldView-2 that can slew 200 km in half that time (DigitalGlobe 2015).

In equation (5) $\omega_{\max}$ is derived from the maneuvering profile, and the first equation controlling the first two seconds of the maneuver is shown in equation (6). For the first two seconds the angular velocity is increased until $\omega_{\max}$ is reached. The following two seconds are at a constant velocity and the remaining two seconds ramp the velocity down. The equations for stage two and stage three of the each maneuver are seen in

Equations (7) and (8), respectively. Additionally, the MATLAB code is available in the Appendix.

$$\omega_{\max} = \begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix} \Big/ 4 \tag{5}$$

For $\mathrm{t_{start}} < \mathrm{t} < \left(\mathrm{t_{start}} + 2\right)$

$$\theta = \frac{\omega_{\max}}{4}\left(t - t_{start}\right)^2 + c \tag{6}$$

For $\left(\mathrm{t_{start}} + 2\right) < \mathrm{t} < \left(\mathrm{t_{start}} + 4\right)$

$$\theta = \omega_{\max}\left(t - \left(t_{start} + 2\right)\right) + \omega_{\max} + c \tag{7}$$

For $\left(t_{start} + 4\right) < t < \left(t_{start} + 6\right)$

$$\theta = \omega_{\max}\left(\mathrm{t} - \left(\mathrm{t_{start}} + 4\right)\right) - \frac{\omega_{\max}\left(\mathrm{t} - \left(\mathrm{t_{start}} + 4\right)\right)^2}{4} + \mathrm{c} + \mathrm{c_2} \tag{8}$$

The equation for the velocity vector, $\dot{\theta}$, is the derivative of the trajectory vector at each state of the maneuver. Equations (9), (10), and (11) show the angular velocity equations used in the model. The calculated angular velocity is used within the model for feed forward control (Sands 2012). Note that this angular velocity is related to the Euler angles and not the angular velocity of the spacecraft body frame. Later, this angular velocity is multiplied by the inertia matrix of the spacecraft and added to the spacecraft velocity vector to calculate the total angular momentum.

For $\mathrm{t_{start}} < \mathrm{t} < \left(\mathrm{t_{start}} + 2\right)$

$$\dot{\theta} = \frac{\omega_{\max}}{2}\left(t - t_{start}\right) \tag{9}$$

For $\left(\mathrm{t_{start}} + 2\right) < \mathrm{t} < \left(\mathrm{t_{start}} + 4\right)$

$$\dot{\theta} = \omega_{\max} \tag{10}$$

For $\left(t_{start} + 4\right) < t < \left(t_{start} + 6\right)$

$$\dot{\theta} = -\frac{\omega_{\max}}{2} t - \left(t_{start} + 4\right) + \omega_{\max} \tag{11}$$

The equation for the acceleration vector, $\ddot{\theta}$, is the derivative of the velocity vector at each state of the maneuver. Equations (12), (13), and (14) show the acceleration equations used in the model. The calculated acceleration is used with angular velocity vector within the model for feed forward control.

For $\mathrm{t_{start}} < \mathrm{t} < \left(\mathrm{t_{start}} + 2\right)$

$$\ddot{\theta} = \frac{\omega_{\max}}{2} \tag{12}$$

For $\left(t_{start} + 2\right) < t < \left(t_{start} + 4\right)$

$$\ddot{\theta} = 0 \tag{13}$$

For $\left(\mathrm{t_{start}} + 4\right) < \mathrm{t} < \left(\mathrm{t_{start}} + 6\right)$

$$\ddot{\theta} = -\frac{\omega_{\max}}{2} \tag{14}$$

These equations are modeled in Simulink within a MATLAB function block; the schematic is shown in Figure 17. The time, Euler angles, and desired slew time enter the command processing block. Within the command processing block there is the index generator and the position-velocity-acceleration function block. This function block can handle an infinite number of consecutive maneuvers, but the index generator that feeds this function block is designed for four maneuvers followed by a return to the starting position.

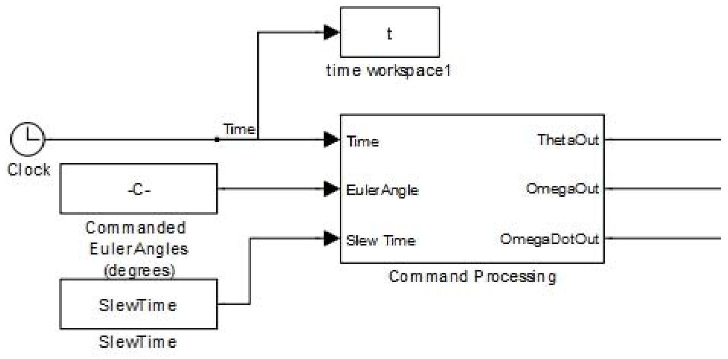


Figure 17. Command processing schematic.

The index generator for the spacecraft model is seen in Figure 18. The index generator could be expanded to handle more maneuvers if necessary. The challenge in passing a parameter argument to MATLAB function blocks is that only read-only constants can be passed; the value cannot come from signals within the Simulink model and must come from variables defined in the MATLAB base workspace. This presents a challenge in dynamically setting an index variable to proceed to the next maneuver. It can be done with data memory blocks and delays on the signal lines but incorporating the memory blocks and signal delays was not done. However, the ability to carry forward the position from one maneuver to the next was incorporated into the model through the calculation of constants in the position equations (6), (7), and (8). The constants (c_x) are calculated using if statements and for-loops during each instantiation of the position vector calculation. The MATLAB code for the calculation of constants can be found in the Appendix. Therefore, the model implicitly assumes successful completion of the last maneuver within the tolerances of the model.

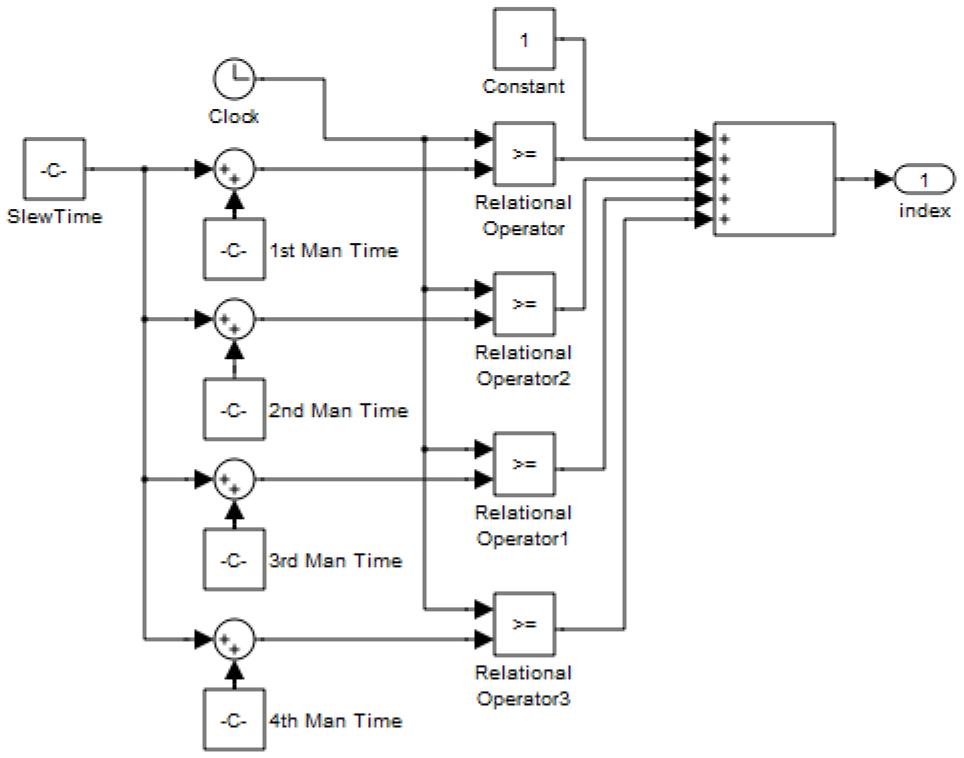


Figure 18. Index generator to queue four maneuvers.

A simple pitch maneuver representing a one degree change in latitude is shown in Figure 19. The coordinate request (37.594788°, -121.876917°), and nadir position (36.59496°, -121.876917°) is loaded into MATLAB, along with the start maneuver time of five seconds. The top plot shows the commanded position versus time. The middle plot shows the velocity versus time and the bottom plot shows the acceleration history.

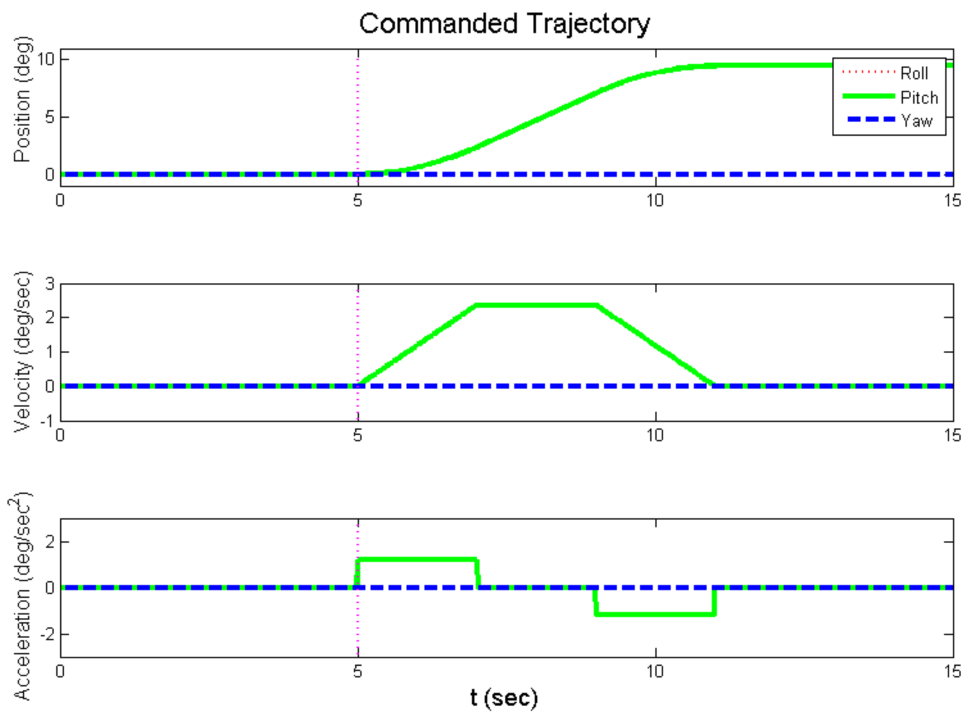


Figure 19. Example slew trajectory profile

Trajectory generation for a spacecraft influences the spacecraft slew behavior and efficiency. The trajectory generator for the spacecraft model used here was designed for smooth transitions to reduce instantaneous velocity changes for the CMGs (Sands 2012). Other trajectories, including time-optimal control maneuvers, could be more effective and efficient for the spacecraft model. Developing a better trajectory generator is discussed in the future work section and is ultimately very important to the implementation of tablet tasking. Assuming that the implementation of satellite tasking via a tablet computer would lead to more imaging requests, the satellite's efficiency would be penultimate in providing a high-level of request fulfillment.

D. CONTROL LAW

The spacecraft model uses proportional-plus-derivative (PD) attitude control. PD control can change the transient response by changing the damping ratio directly (M. Driels 1996). The Euler angles from the spacecraft are subtracted from the desired Euler angles and input into the PD controller. The PD controller was tuned to a K_p value of 50 and a K_d value of 1000. The transfer equation for the PD Controller is from MATLAB, and is shown in (15). In the compensator formula shown, P is the proportional variable, D the derivative value, and N is the filter coefficient, where $N = 100$. The PD controller selected is labeled with a 1 in Figure 20. The output from the PD controller feeds into the CMG function block.

$$P\left(1 + D\frac{N}{1 + N\frac{1}{S}}\right) \tag{15}$$

The spacecraft model also uses feedforward control. The feedforward control uses the commanded trajectory and sends a torque control signal directly to the system. The feedforward controller is open-loop control and must be used in conjunction with the PD control. The insertion of the feedforward control is labeled with a 2 in Figure 20. The primary benefit of the feedforward controller is to reduce the time lag associated with the feedback controller.

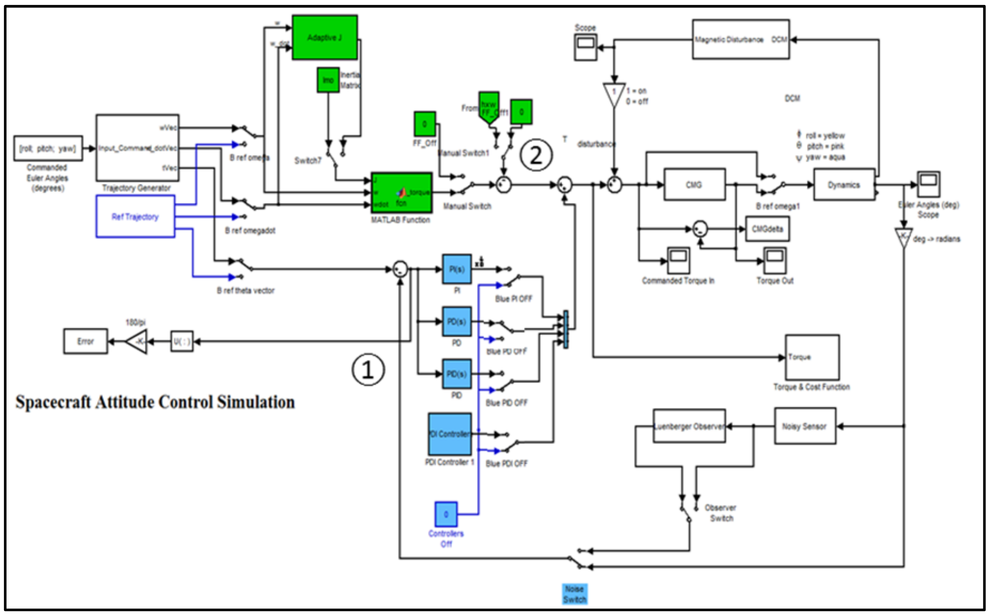


Figure 20. Control block diagram in SIMULINK.

E. CMGS

The spacecraft model includes CMGs because this is the hardware configuration of available test beds at NPS. The three-axis spacecraft simulator, (TASS2) and the Dark Mirror testbed both utilize CMGs. Since the next obvious progression in proving this concept is to use one of these available test beds the spacecraft simulator was designed with CMGs. A secondary reason is because of the prominent use of CMGs on future imaging spacecraft, including WorldView-1, WorldView-2 and WorldView-3.

The simulated CMGs are modeled as if they were arranged in a standard pyramid as shown in Figure 21.

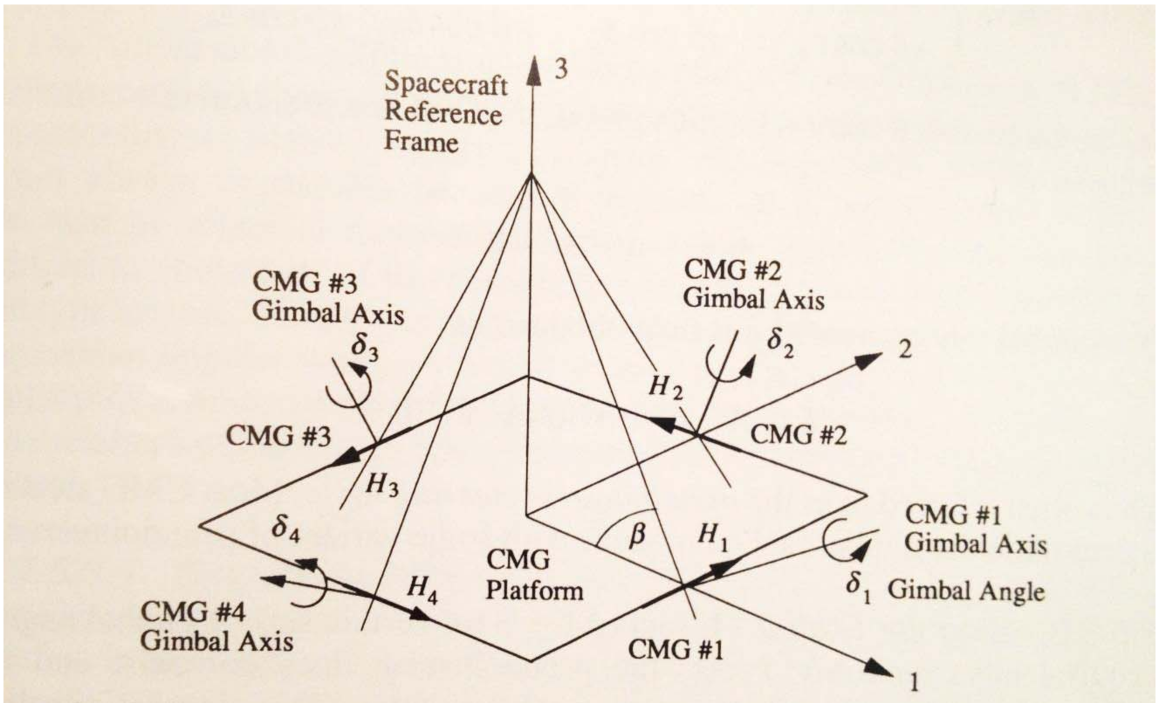


Figure 21. CMG configuration for spacecraft model from Wei, Space Vehicle Dynamics and Control Second Edition 2008.

The total CMG angular momentum vector, in the spacecraft reference frame is expressed in Equation (15). The four CMGs each with a skew angle, β, of 54.73 degrees and an initial gimbal angle of 0 degrees so that the angular momentum is zero at the beginning of the simulation.

$$\mathbf{h} = \begin{bmatrix} -\cos\beta\sin\delta_1 \\ \cos\delta_1 \\ \sin\beta\sin\delta_1 \end{bmatrix} + \begin{bmatrix} -\cos\delta_2 \\ -\cos\beta\sin\delta_2 \\ \sin\beta\sin\delta_2 \end{bmatrix} + \begin{bmatrix} \cos\beta\sin\delta_3 \\ -\cos\delta_3 \\ \sin\beta\sin\delta_3 \end{bmatrix} + \begin{bmatrix} \cos\delta_4 \\ -\cos\beta\sin\delta_4 \\ \sin\beta\sin\delta_4 \end{bmatrix} \tag{15}$$

The desired CMG torque, $\dot{\mathbf{h}}$, is determined using Equation (16). In this equation, $\mathbf{u}$ is the commanded control torque input (from the control law), and $\boldsymbol{\omega}$ the spacecraft angular velocity vector.

$$\dot{\mathbf{h}} = -\mathbf{u} - \boldsymbol{\omega} \times \mathbf{h} \tag{16}$$

Next, the gimbal rate command can be determined using the following equation

$$\dot{\boldsymbol{\delta}} = \mathbf{A}^{+}\dot{\mathbf{h}} = \mathbf{A}^{\mathrm{T}}\left(\mathbf{A}\mathbf{A}^{\mathrm{T}}\right)^{-1}\dot{\mathbf{h}} \tag{17}$$

where

$$\mathbf{A} = \frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}} = \begin{bmatrix} -\cos\beta\cos\delta_1 & \sin\delta_2 & \cos\beta\cos\delta_3 & -\sin\delta_4 \\ -\sin\delta_1 & -\cos\beta\cos\delta_2 & \sin\delta_3 & \cos\beta\cos\delta_4 \\ \sin\beta\cos\delta_1 & \sin\beta\cos\delta_2 & \sin\beta\cos\delta_3 & \sin\beta\cos\delta_4 \end{bmatrix} \tag{18}$$

The SIMULINK CMG schematic used in the spacecraft model is shown in Figure 22. The function blocks execute the equations for pseudoinverse steering logic. The Pseudoinverse block takes the matrix $\mathbf{A}$ and creates the Moore-Penrose Pseudoinverse, ($\mathbf{A}^{+}$) that satisfies the identities shown in equation (17). The commanded control torque input, $\mathbf{u}$, enters the schematic from the left and the CMG torques are output to the dynamics block of the spacecraft model described in the next section.

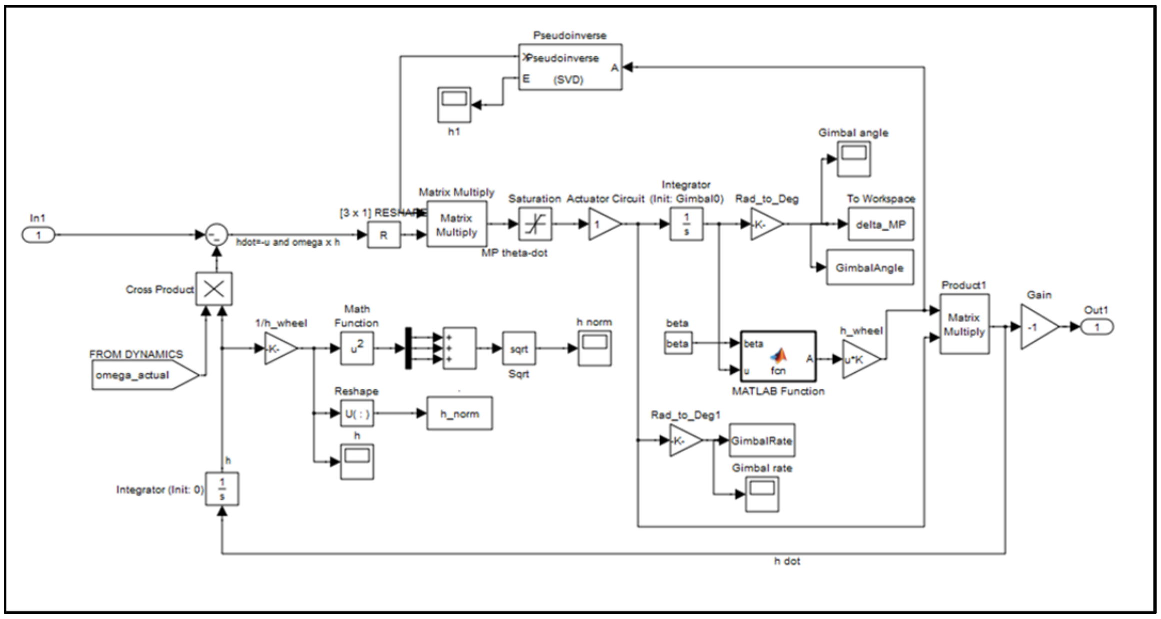


Figure 22. CMG schematic from SIMULINK spacecraft model.

F. SATELLITE ATTITUDE DYNAMICS

The spacecraft attitude dynamics and kinematics are modeled to determine the angular velocity and attitude of the satellite with respect to the earth. A schematic of the dynamics is found in Figure 23.

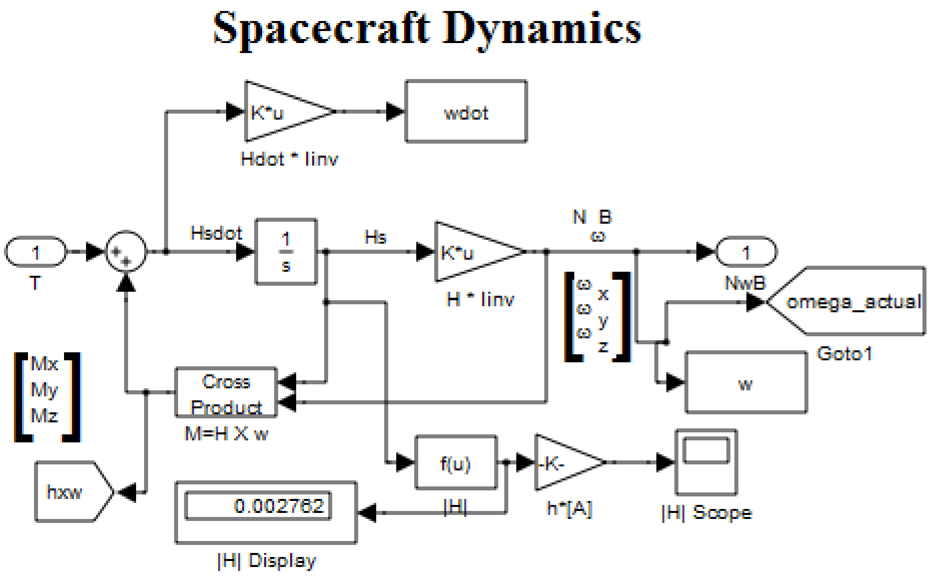


Figure 23. Spacecraft dynamics schematic from SIMULINK spacecraft model

Within the dynamics block of the SIMULINK model, the torque vector, $\mathbf{T}$, is received from the CMG subsystem. It is summed with spacecraft momentum, $\mathbf{M}$, the cross product of $\mathbf{H}_s$ and $\boldsymbol{\omega}$, yielding $\dot{\mathbf{H}}_s$, as seen in Equation (19) .

$$\dot{\mathbf{H}}_s = \mathbf{H}_s \times \boldsymbol{\omega} + \mathbf{T} \tag{19}$$

The derivative of angular momentum vector, $\dot{\mathbf{H}}_s$, is then integrated and multiplied by the inverse of the inertia matrix $\mathbf{I}_{\mathbf{inv}}$ (see Equation (20) to Equation (22)) (Sidi 2006). The angular velocity vector is then passed to the kinematics block.

$$\mathbf{I}_{\mathbf{inv}} = \begin{bmatrix} 0.0085 & 0.0008 & 0.0003 \\ 0.0008 & 0.0069 & -0.0014 \\ 0.0003 & -0.0014 & 0.0097 \end{bmatrix} \tag{20}$$

$$\mathbf{H}_s * \mathbf{I}_{inv} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{21}$$

$$\mathbf{H}_s = \mathbf{I}\boldsymbol{\omega} \tag{22}$$

G. SATELLITE KINEMATICS

The rotational rate , $\boldsymbol{\omega}$,of the spacecraft (the body with respect to inertial) enters the kinematics block of the model, $(\omega_x,\ \omega_y,\ \omega_z)$ and the $\boldsymbol{\omega}_{orbit}$ of the orbit is subtracted off (Equation (23)). The $\mathbf{DCM}$ will be defined later. The result is the angular rate of the body with respect to orbit $(\omega_1,\ \omega_2,\ \omega_3)$.

$$\boldsymbol{\omega}_{orbit} = diagonal(\mathbf{DCM}) * \sqrt{\frac{\mu}{(\mathrm{R}_E + h)^3}} \tag{23}$$

From spacecraft $\boldsymbol{\omega}$, with respect to orbit, the quaternions are calculated. The block diagram of the kinematics can be seen in Figure 24.

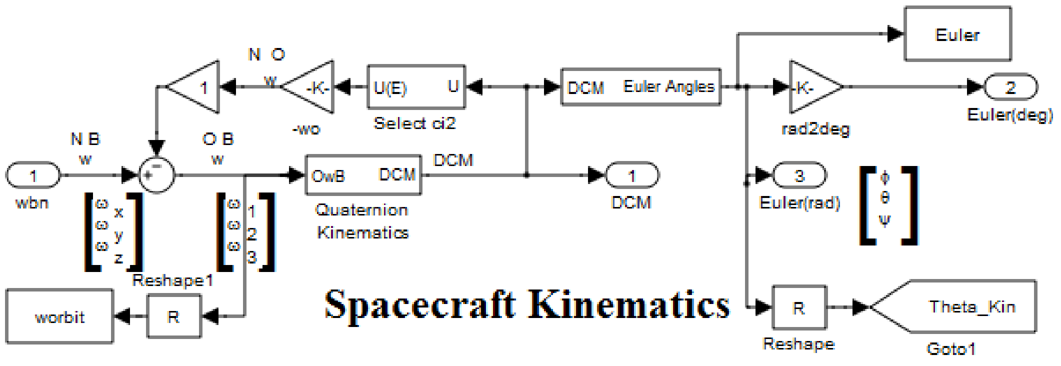


Figure 24. Spacecraft kinematics schematic from SIMULINK spacecraft model.

The quaternions are calculated to avoid singular situations (divide by zero errors) inherent to an Euler angle attitude parameterization. The governing system of equations for determining the quaternion of the satellite given $\boldsymbol{\omega}$ is shown in Equation (24).

$$\dot{q} = \frac{1}{2}\begin{bmatrix} q_4 & -q_3 & q_2 & q_1 \\ q_3 & q_4 & -q_1 & q_2 \\ -q_2 & q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ 0 \end{bmatrix} \tag{24}$$

The equation to convert from the quaternion system to the DCM is given by Equation (25). Assuming the rotation from an arbitrary body axis to the principle axis is a yaw-pitch-roll rotation, and the expanded DCM to convert to Euler Angles is given in Equation (26).

$$\mathbf{DCM} = \begin{bmatrix} 1-2\left(\boldsymbol{q}_2^2+\boldsymbol{q}_3^2\right) & 2\left(\boldsymbol{q}_1\boldsymbol{q}_2+\boldsymbol{q}_3\boldsymbol{q}_4\right) & 2\left(\boldsymbol{q}_1\boldsymbol{q}_3-\boldsymbol{q}_2\boldsymbol{q}_4\right) \\ 2\left(\boldsymbol{q}_2\boldsymbol{q}_1-\boldsymbol{q}_3\boldsymbol{q}_4\right) & 1-2\left(\boldsymbol{q}_1^2+\boldsymbol{q}_3^2\right) & 2\left(\boldsymbol{q}_2\boldsymbol{q}_3+\boldsymbol{q}_1\boldsymbol{q}_4\right) \\ 2\left(\boldsymbol{q}_3\boldsymbol{q}_1+\boldsymbol{q}_2\boldsymbol{q}_4\right) & 2\left(\boldsymbol{q}_3\boldsymbol{q}_2-\boldsymbol{q}_1\boldsymbol{q}_4\right) & 1-2\left(\boldsymbol{q}_1^2+\boldsymbol{q}_2^2\right) \end{bmatrix} \tag{25}$$

$$\mathbf{DCM} = \begin{bmatrix} \cos(\varphi)\cos(\psi) & \cos(\varphi)\sin(\psi) & -\sin(\varphi) \\ \sin(\theta)\sin(\varphi)\cos(\psi)-\cos(\theta)\sin(\psi) & \sin(\theta)\sin(\varphi)\sin(\psi)-\cos(\theta)\cos(\psi) & \sin(\theta)\cos(\varphi) \\ \cos(\theta)\sin(\varphi)\cos(\psi)+\sin(\theta)\sin(\psi) & \cos(\theta)\sin(\varphi)\cos(\psi)-\sin(\theta)\cos(\psi) & \cos(\theta)\cos(\varphi) \end{bmatrix} \tag{26}$$

The **DCM** is calculated from the quaternions, and the 3x3 matrix is an output of the kinematics block. The block diagram of the quaternion calculations and direction cosine matrix (DCM) can be seen in Figure 25.

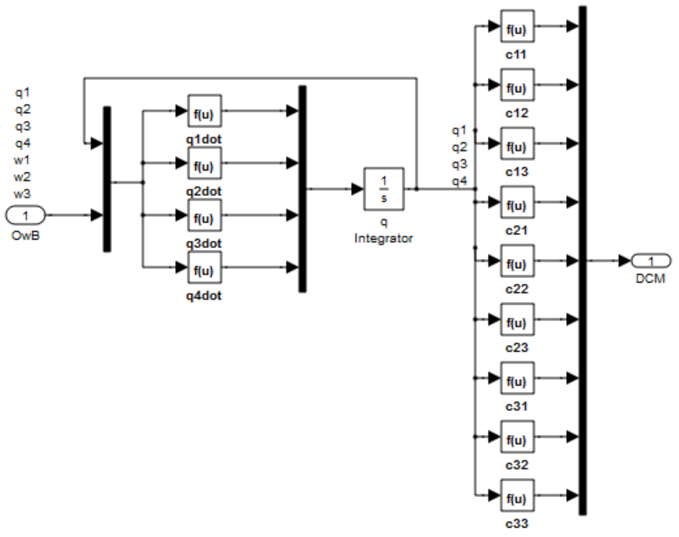


Figure 25. Quaternion and direction cosine matrix SIMULINK diagram.

The output of the kinematics block goes into the disturbance blocks and is also fed back into the controller. The disturbance blocks are selectable and can be turned on or off. For the purposes of this satellite tasking demonstration the magnetic, aero, solar and gravity gradient disturbances are turned off.

H. MODEL VERIFICATION AND VALIDATION

1. Command Processing and Trajectory Generation

The first step in verifying the spacecraft model was to compare the input latitudes and longitudes to the roll and pitch angles. Once satisfied that the attitude vectors generated by the model matched the hand calculations, they were compared to the velocity and acceleration vectors. A plot showing this is located previously in Figure 19.

2. CMG Model Verification

The next step was to verify the CMG block in the simulation. The commanded trajectory was plotted next to the gimbal rates in Figure 26 to see what was happening over time. The top plot shows the commanded trajectory from the first slew, and the four plots below show the individual rates for the gimbals. A best effort was made to smooth the trajectory, there are unrealistic gimbal rate changes seen in this plot. Instantaneous rate changes occur primarily when the trajectory transitions from a quadratic slope to the linear slope, approximately two and four seconds after the start of the maneuver (at approximately 7 and 9 seconds on the plots). This is unrealistic for physical hardware but the performance is suitable for the purposes of this simulation.

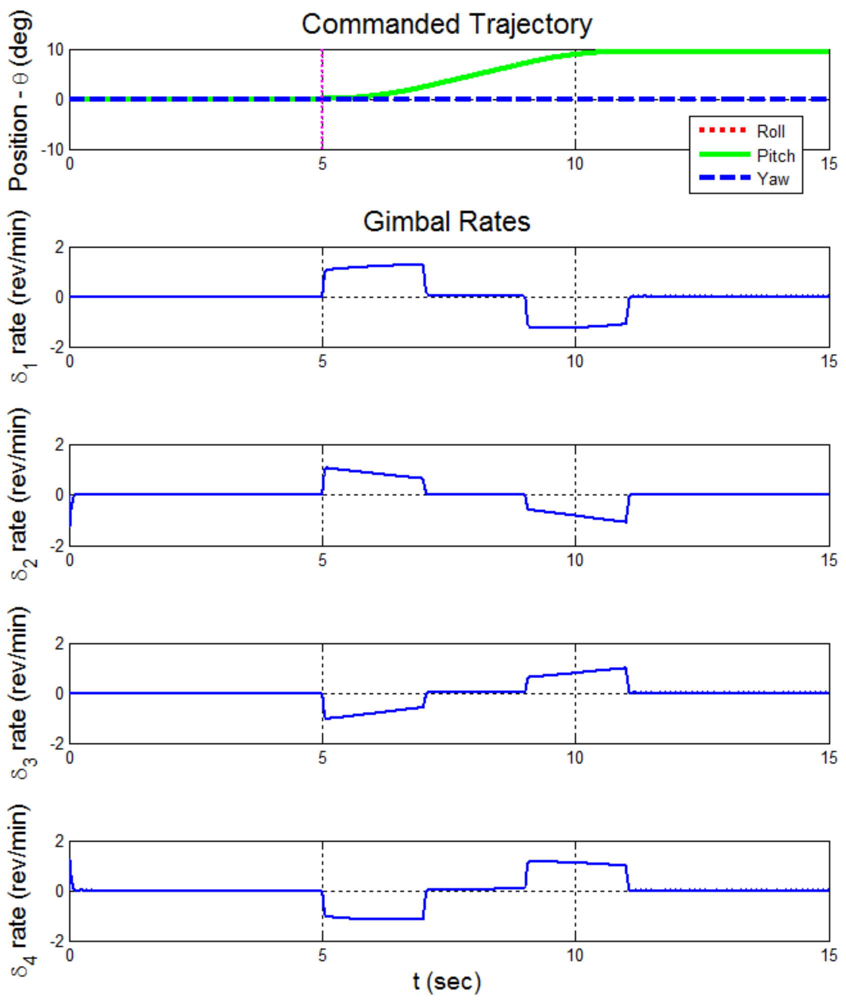


Figure 26. Commanded trajectory and the rates of the four gimbals.

Following the analysis of the rates, the gimbal angles were plotted, see Figure 27. The gimbal angles change accordingly to the rates seen in Figure 26. This step was to check for sign errors and for realistic maneuvering of the gimbals.

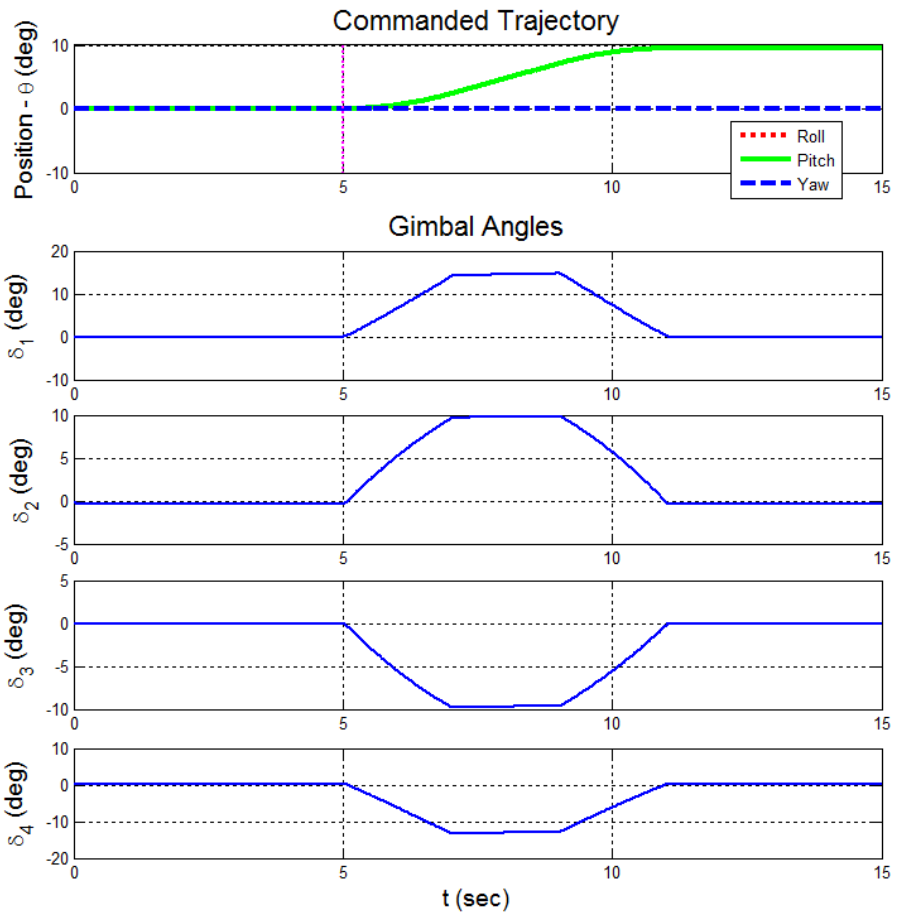


Figure 27. Commanded trajectory and four gimbal angles.

3. Kinematic Model Verification

The rotational rate of the spacecraft, the body with respect to inertial, is plotted in Figure 28. This figure shows the relationship between the spacecraft axis and roll pitch and yaw. Velocity is seen in the *x*-direction when the spacecraft rolls, and it is seen in the *y*-direction when the spacecraft pitches. For the example roll and pitch maneuver, velocity is seen in *x*, *y* and *z*-directions. The velocity seen in the *z*-direction are a result of coupling from the roll and pitch maneuvers.

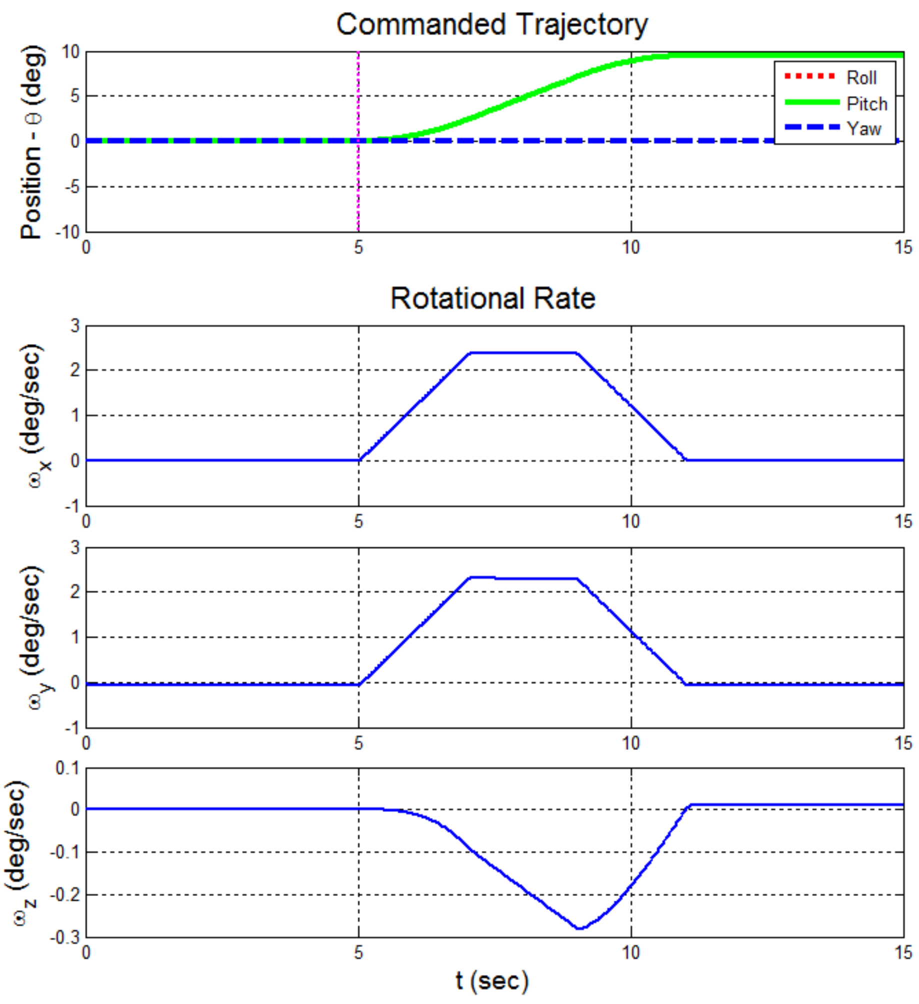


Figure 28. Rotational rate of the spacecraft plotted with the commanded trajectory.

As part of verification and validation of the SIMULINK model the values of q_1, q_2, q_3, and q_4 from the dynamics block were squared and summed for each time-step of the simulation. Plotting the value of this variable for the duration of the simulation showed that the sum of the quaternions equaled one the entire time. A plot of the quaternions is provided in Figure 29.

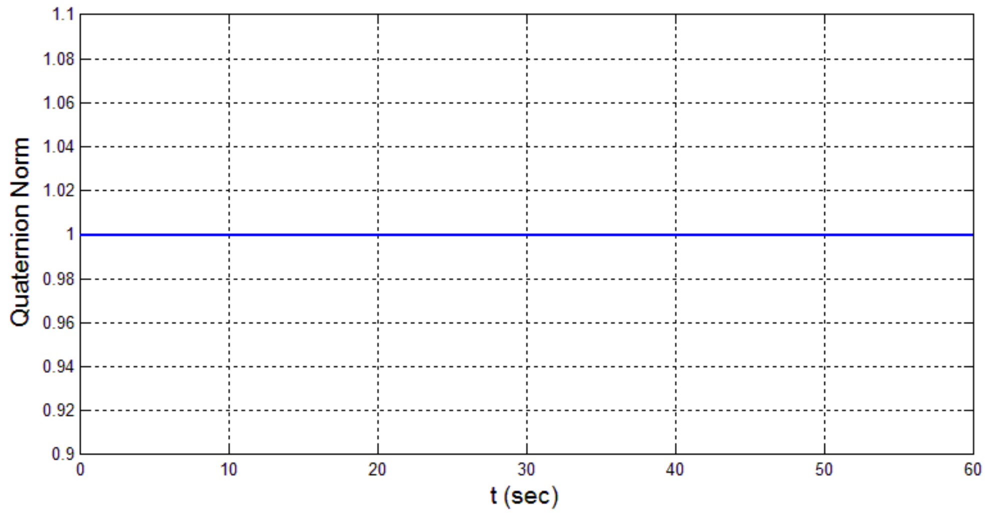


Figure 29. Quaternion norm for the example maneuver.

A second validation and verification of the SIMULINK spacecraft model required the plotting the simulated spacecraft angular momentum and the CMG angular momentum. The expected result is that the sum of the angular momentums equals zero due to conservation of angular momentum as seen in Figure 30.

.

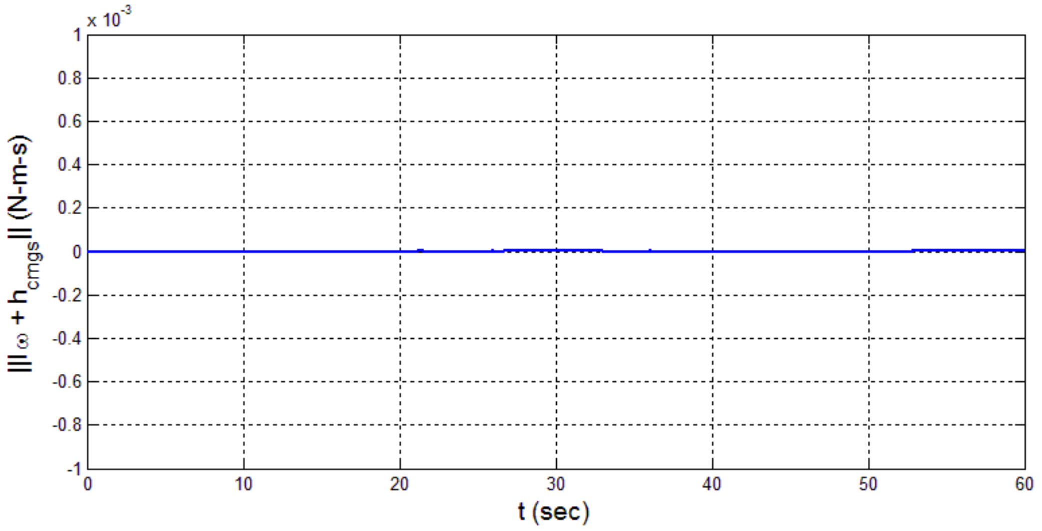


Figure 30. Momentum conservation for example maneuver.

I. SUMMARY

This chapter provided an overview of the spacecraft model used with the satellite tasking application, details of the implementation, and examples of how the model was validated. The model takes an input of coordinates from the TAST application and processes them into roll, pitch and yaw angles. These angles are then made into Euler angles for the spacecraft trajectory. The trajectory is followed by the spacecraft model as shown in the plots throughout the chapter. The model also uses a PD and feedforward control to reduce the time lag of the feedback controller. The validation section of this chapter shows more plots to visualize how future implementation on the test bed would appear. Finally, the analysis of the simulation showed that $\| \mathbf{q} \| = 1$ and $\| \mathbf{I}_{\boldsymbol{\omega}} + \mathbf{h}_{\mathbf{CMG}} \| = 0$ as required. The following chapter includes a demonstration of the iPad tasking application (described in Chapter II) interacting with the spacecraft model of the previous chapter.

IV. DEMONSTRATION OF TABLET TASKING

A. INTRODUCTION

This chapter demonstrates the tasking application with the model spacecraft. The user selects four sample points on the iPad, and sends a request for imagery products. The request is processed and the model satellite slews to collect the requested areas. Data from the slews is returned to the application on the iPad. The data returned is a placeholder for image products that would be returned from a hardware testbed or actual satellite.

B. SAMPLE REQUEST

As an example use case, four different points are requested through the TAST application running on the iPad. A satellite image of the four points (B-E) and the sub-satellite point (A) is presented in Figure 31. The coordinates and associated target labels are indicated in Table 4.

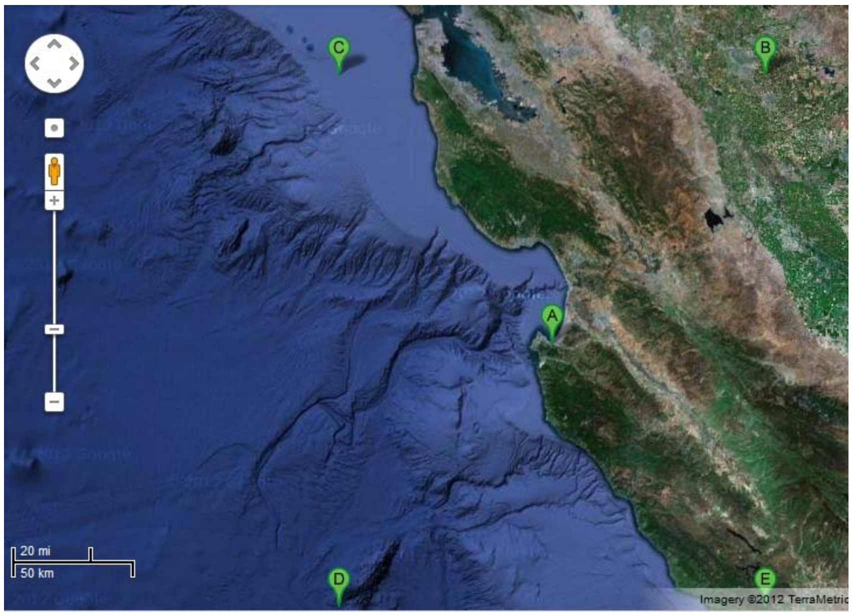


Figure 31. Sample points requested for collection.

Number	Target Label	Latitude (deg)	Longitude (deg)	Slew Start Time (sec)
	Starting Location	36.59496	-121.876917	
1	Target B	37.594788	-120.876924	5
2	Target C	37.594788	-122.876924	18
3	Target D	35.594788	-122.876924	25
4	Target E	35.594788	-120.876924	33

Table 4. Sample coordinates and associated target labels for scenario of Figure 31.

The requests are then downloaded from the server and loaded into the MATLAB/SIMULINK project with an m-file script. The m-script file parses the request data, calculates the slew angles, and launches the spacecraft simulation.

C. SAMPLE SPACECRAFT SIMULATION

A sample spacecraft simulation of the four requested maneuvers with a return to the original position can be seen in Figure 32. The top graph in the figure is the commanded attitude plotted against the time. Roll is represented by a red dashed line, pitch by a solid green, and yaw with a dashed blue line. The start times for each maneuver are depicted by the dotted lines seen at times 5, 18, 25, 33, and 44 seconds. At five seconds the spacecraft position trajectory increases for the first two seconds, followed by steady movement, and finishing the command with a slow-down leveling off at the desired position, approximately nine degrees in roll and pitch. This position is held until the second start time is reached at 18 seconds. At 18 seconds, the spacecraft is commanded to roll approximately 18 degrees. This process continues for the next two commands and the return to origin. The middle plot gives the velocity components plotted verse time. At five seconds, the velocity increases until the maximum angular velocity for the commanded position is reached. The velocity holds for two seconds and then declines steadily over the next two seconds. The spacecraft is given time to settle and is held steady between the requests to emulate imaging. The roll, pitch, and yaw velocities are seen in the center plot of Figure 32. At around 35 seconds the maneuver has a rate of five degrees per second, this is greater than the Worldview-2 max rate.

Below the velocities, the roll, pitch and yaw accelerations are also plotted. These three plots depict how the requested coordinates, previously translated into slew angles, are interpreted into instructions for the spacecraft attitude control system.

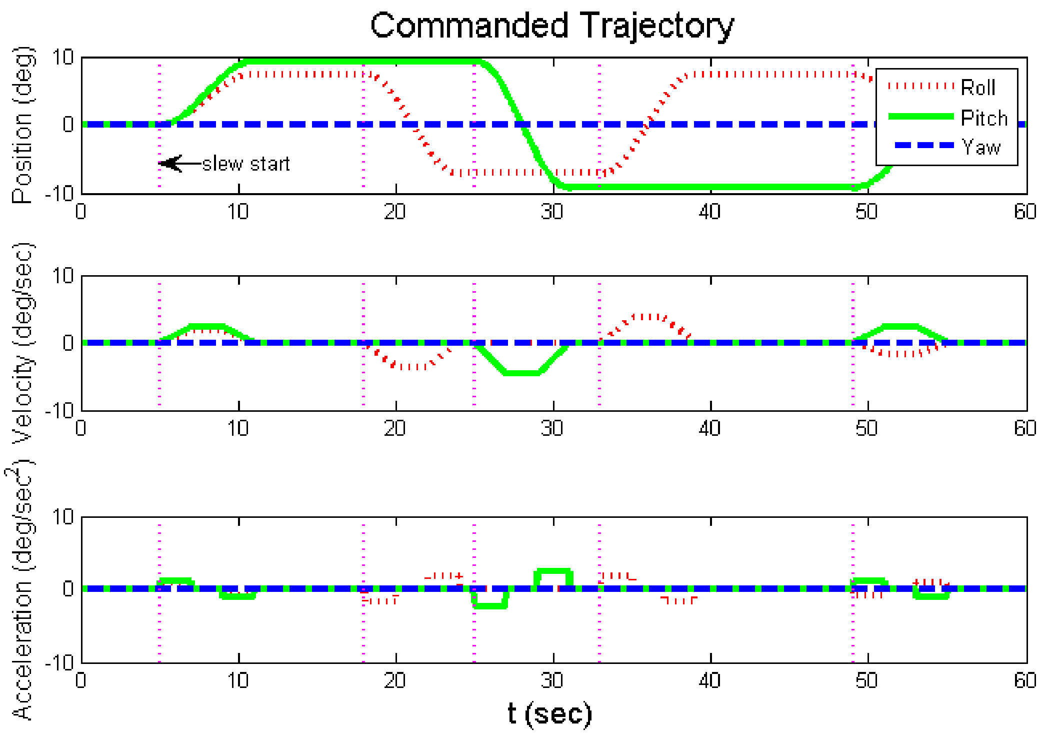


Figure 32. Output of spacecraft trajectory generator.

Number	Target Label	Roll (deg)	Pitch (deg)	Yaw (deg)
	Starting Location	0	0	0
1	Target B	7.3797	9.2835	0
2	Target C	-7.1427	9.2835	0
3	Target D	-7.1427	-9.2202	0
4	Target E	7.3797	-9.2202	0

Table 5. Desired roll, pitch, and yaw angles for each target.

The commanded positions enter the CMG subsystem block and the gimbal rates ($\dot{\delta}$) for the CMGs are computed as per the feedback/feedforward logic described in Chapter III. The commanded gimbal rates are shown in Figure 33. The top plot is the commanded trajectory for reference, and the four plots below it are the rates for each one of the four CMGs.

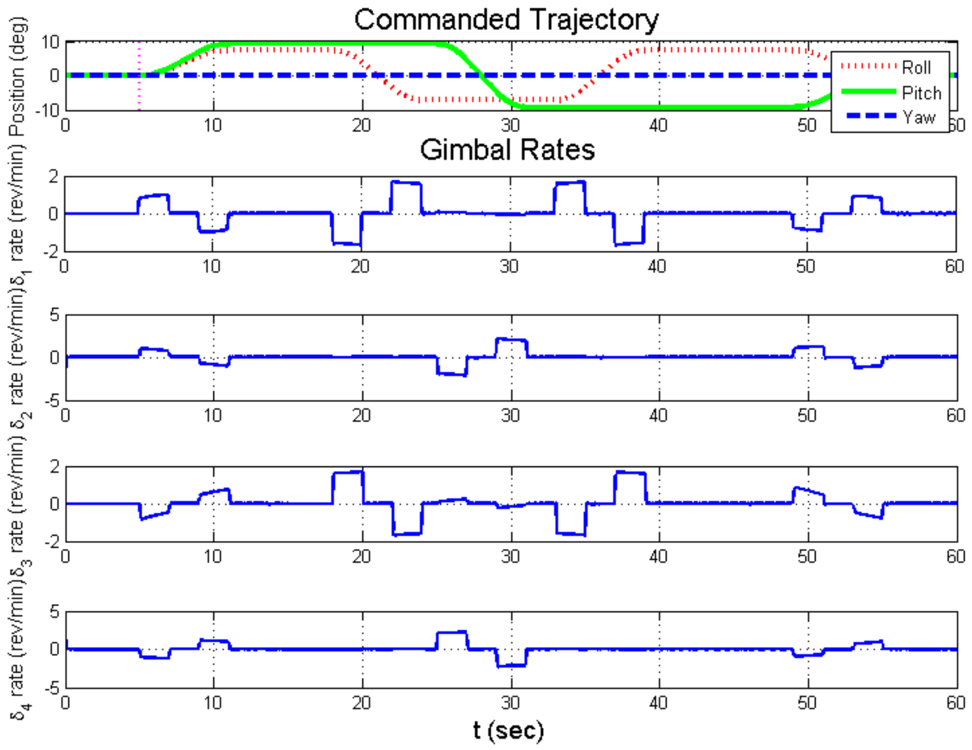


Figure 33. Commanded trajectory and rates of the CMG gimbals.

Related to the gimbal rates are the gimbal angles. The gimbal angles for the four maneuvers are shown in Figure 34. Again, the attitude command trajectory is shown in the top plot for reference and the angles of the four CMGs are plotted below. The gimbal angles are shown to demonstrate the gimbal angles exercised for these example maneuvers. The angles seen in these maneuvers could serve as a starting point for the development of hardware requirements if future work of incorporating a test bed is executed.

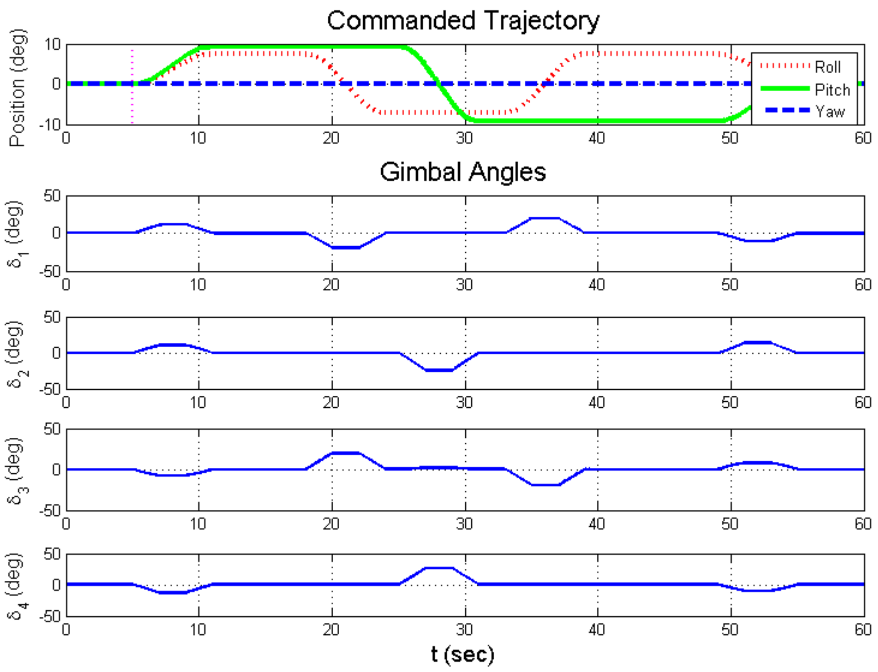


Figure 34. Commanded trajectory and gimbal angles for four sample maneuvers

The rotational rate of the spacecraft, the rotation of the body with respect to the inertial frame, is plotted in Figure 35. The figure shows the relationship between the spacecraft axis and roll pitch and yaw. Velocity is seen in the *x*-direction when the spacecraft rolls, and it is seen in the *y*-direction when the spacecraft pitches. When the spacecraft rolls and pitches simultaneously coupling is seen is the *z*-direction. In this simulation when the spacecraft is executing a large pitch maneuver a significant rate is seen in the *z*-direction at about 28 seconds. This happens because the example spacecraft has a dense inertia tensor, similar to the testbed. The spacecraft's initial axis of rotation is not entirely quiescent and when a large pitch maneuver is executed angular momentum is conserved causing some rotation about the *z*-direction.

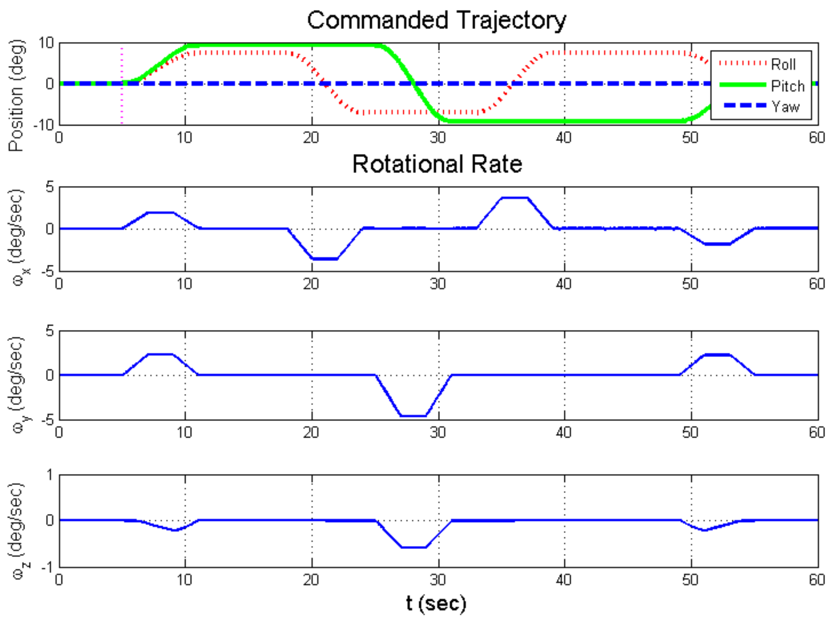


Figure 35. Rotational rate of the spacecraft plotted with the commanded trajectory

The Euler angle error is plotted in degrees in Figure 36. It is the commanded trajectory with the feedback Euler angles from the dynamics block subtracted from it. The PD controller drives the steady state error to zero after the slews complete and before the imaging phase. The Euler angle errors demonstrate the accuracy of the slew maneuvers and are related to the overall pointing error of the spacecraft. The next section discusses the output of the spacecraft simulation.

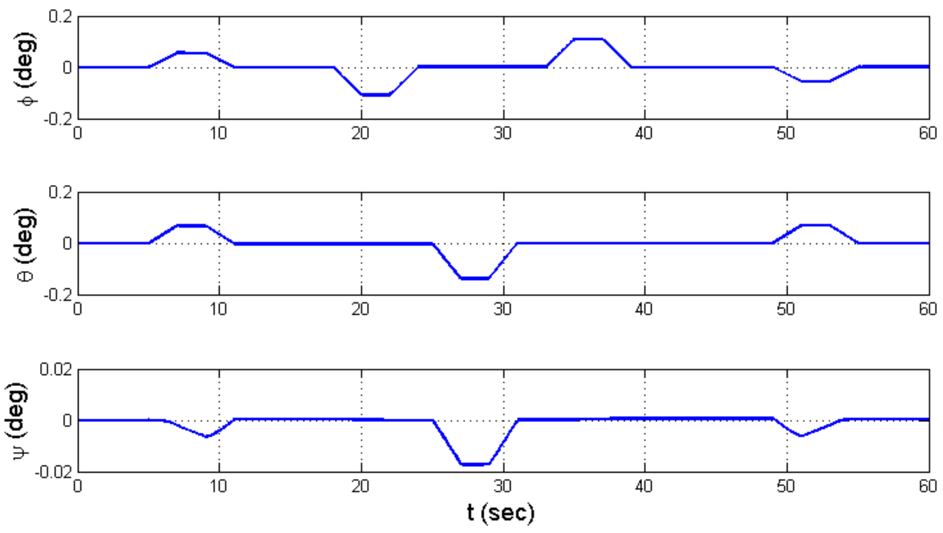


Figure 36. Euler angle error for the four slews with the PD controller.

D. OUTPUT DATA

Once the simulated image has been taken by the satellite, the data must be downlinked to a terrestrial server or routed through communication satellites to the tablet. In this proof of concept, the error associated with pointing is returned to the server instead of images. The MATLAB code outputs the error values to a text file and the text file is uploaded to the server so that it can be retrieved by the tablet application.

The first step in computing the pointing error in the simulation is to find the index associated with the end of the maneuver. One needs to judge when a practical time would be for the sensor to take the image; either immediately after the slew is completed or sometime afterward. In this example, the slew is given six seconds to complete, and then an additional two seconds of settling time. The reference trajectory and Euler angles are compared at the selected index time. The pointing error is reported in both the Euler angle error and the associated latitude and longitude coordinates. The latitude and longitude coordinates are calculated according to Equations (27) and (28).

$$\Delta\phi = \tan(\phi) * h_{orb} \tag{27}$$

$$\Delta\lambda = \tan(\theta) * h_{orb} \tag{28}$$

The resulting coordinates are then compared the desired coordinates and using the Haversine formula the distance between the two points on Earth is calculated, Equations (29)-(32). The formula does not take into account the oblate shape of the Earth and uses a value for the Earth's radius of 6378 km (Robusto 1957). The oblate shape of the earth affects the latitude more than the longitude. Most references use the geodetic latitude, and since the difference between the geocentric latitude is slight it is also used in this thesis (Bate, Mueller and White 1971).

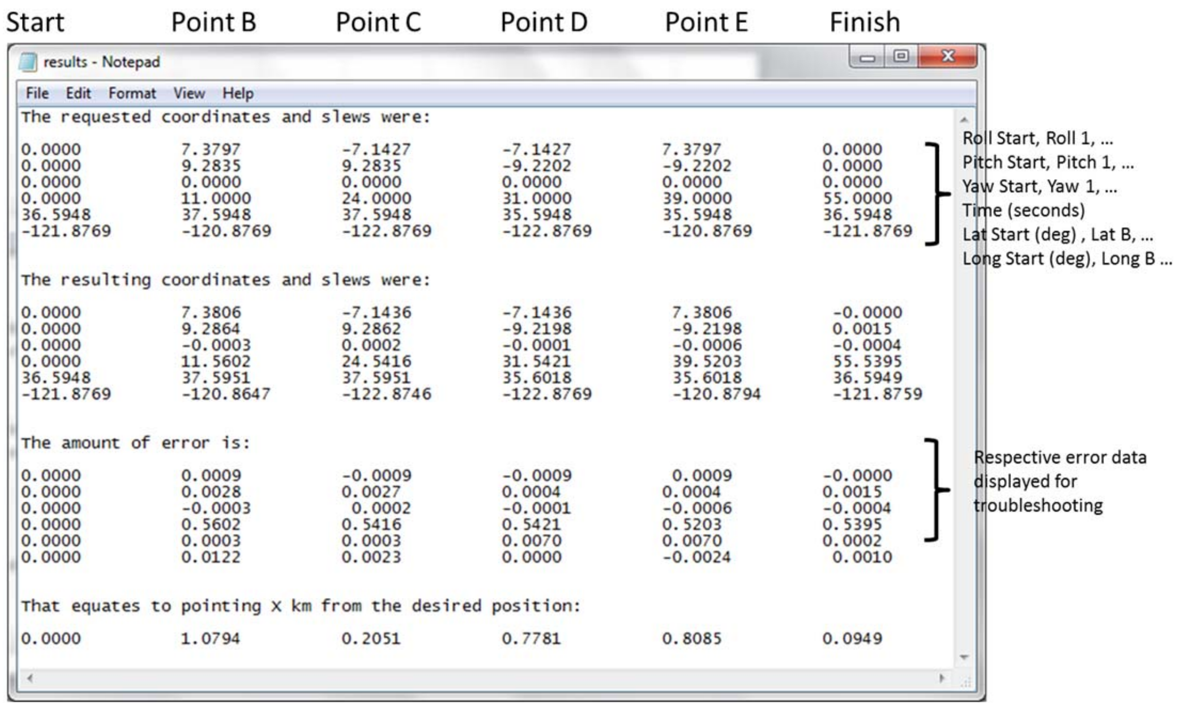
$$\begin{aligned} \Delta\phi &= \left(\phi_{\mathrm{f}} - \phi_{\mathrm{s}}\right) \\ \Delta\lambda &= \left(\lambda_{\mathrm{f}} - \lambda_{\mathrm{s}}\right) \end{aligned} \tag{29}$$

$$\mathrm{a} = \sin\left(\frac{\Delta\phi}{2}\right)^2 + \cos\phi_{\mathrm{s}} * \cos\phi_{\mathrm{f}} * \sin\left(\frac{\Delta\lambda}{2}\right)^2 \tag{30}$$

$$\mathrm{c} = 2 * \arctan\left(\frac{\sqrt{\mathrm{a}}}{\sqrt{1-\mathrm{a}}}\right) \tag{31}$$

$$\mathrm{d} = \mathrm{R}_{\mathrm{E}} * \mathrm{c} \tag{32}$$

This data is output to a text file named results.txt. A screenshot of the output file is shown in Figure 37. The folder on the laptop is setup to be synched with the folder on the server http://184.169.151.216/results/results.txt. From that location on the server the PHP-script automatically pulls the current data into the file that the iPad application can access and load for the user to view. This output file from MATLAB is analogous to the image data that the spacecraft would actually send back to the user. The pointing error comes from several sources, but the biggest contributor is from the model's estimation of distances and angles between the coordinates.



Start
Point B
Point C
Point D
Point E
Finish
results - Notepad
File Edit Format View Help
Roll Start, Roll 1, ...
Pitch Start, Pitch 1, ...
Yaw Start, Yaw 1, ...
Time (seconds)
Lat Start (deg) , Lat B, ...
Long Start (deg), Long B ...
Respective error data displayed for troubleshooting

```
The requested coordinates and slews were:

0.0000      7.3797      -7.1427     -7.1427     7.3797      0.0000
0.0000      9.2835      9.2835      -9.2202     -9.2202     0.0000
0.0000      0.0000      0.0000      0.0000      0.0000      0.0000
0.0000      11.0000     24.0000     31.0000     39.0000     55.0000
36.5948     37.5948     37.5948     35.5948     35.5948     36.5948
-121.8769   -120.8769   -122.8769   -122.8769   -120.8769   -121.8769

The resulting coordinates and slews were:

0.0000      7.3806      -7.1436     -7.1436      7.3806      -0.0000
0.0000      9.2864      9.2862      -9.2198      -9.2198     0.0015
0.0000      -0.0003     0.0002      -0.0001      -0.0006     -0.0004
0.0000      11.5602     24.5416     31.5421      39.5203     55.5395
36.5948     37.5951     37.5951     35.6018      35.6018     36.5949
-121.8769   -120.8647   -122.8746   -122.8769    -120.8794   -121.8759

The amount of error is:

0.0000      0.0009      -0.0009     -0.0009      0.0009     -0.0000
0.0000      0.0028      0.0027      0.0004      0.0004      0.0015
0.0000      -0.0003      0.0002     -0.0001     -0.0006     -0.0004
0.0000      0.5602      0.5416      0.5421      0.5203      0.5395
0.0000      0.0003      0.0003      0.0070      0.0070      0.0002
0.0000      0.0122      0.0023      0.0000      -0.0024      0.0010

That equates to pointing X km from the desired position:

0.0000      1.0794      0.2051      0.7781      0.8085      0.0949
```

Figure 37. MATLAB/SIMULINK data output file.

E. SUMMARY OF DEMONSTRATION

This chapter demonstrated an example request from the TAST application, the processing of the request into commands for a model spacecraft, the execution of the maneuvers on the simulated spacecraft and the return of data back to the user (in terms of a data file that reports the imaging attitude errors). The four example image requests of the user were chosen to represent a practical request to view the terrain (from satellite imagery) in different directions. The scenario was meant to depict a user that wanted near-real-time satellite imagery of the surrounding area. The locations were not particularly far apart and show a reasonable use case. However, in running through the scenario, the slew rates were in excess of the capabilities of commercial satellites currently in orbit. This issue has many solutions. One solution would be to increase slew times of the model, or put limits on the user requests. A more expansive solution would be to increase the slew rate capabilities of future satellites. Either way this demonstration of the TAST application provides a basis for pursuing the concept of giving the warfighter a near-real-time view of enemies behind the hill at their request.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION AND FUTURE WORK

A. CONCLUSION

This presented a proof of concept application for a tablet tasking of a satellite. The tablet application called TAST was coded to first submit requests to a server. Once on the server, the requests are downloaded and the spacecraft simulation in SIMULINK executes the requests. Once the imaging process is complete, a text file with error metrics is created and uploaded back to the server for the application to retrieve. The text file is intended to be analogous to the image file sent from the spacecraft back to the user.

Throughout the thesis there are areas of research and development that were only cursorily addressed as they did not relate directly to proving the concept. The following sections address areas that could be follow-on work in support of further developing the tablet application.

B. FUTURE WORK

1. Spacecraft Model

There are many spacecraft models available in the public domain and for students. The spacecraft model used to evaluate the tablet application was the best model that could be created in the given timeframe. Improvements could be made to the model created for this proof of concept, or the tablet application's outputs could be input into another tasking and spacecraft model.

One particular area that could be improved is the trajectory generation for the roll, pitch and yaw commands. Different trajectories could be used to improve the performance and reduce error. The trajectories used here were not optimized.

The SIMULINK model included coding for magnetic torque disturbances, along with aero and solar disturbances. The torque created by the gravity gradient of the spacecraft is also included. These disturbances would not have had an effect on the model for the short period of time that the simulation was run to demonstrate the tasking capabilities of the tablet. Therefore, the disturbances model was disabled. However, activating these disturbances could be useful for longer demonstrations and in providing a higher fidelity model of the spacecraft dynamics.

The model also assumed perfect observers for the spacecraft Euler angles, but the SIMULINK model includes switches to insert noisy sensors and a Luenberger Observer if desired. Again, these additions to the model could be useful in future work and in creating a higher fidelity, longer running model.

2. Tasking Application Architecture

The subject of human computer interaction is very broad and this thesis only scratched the surface on the applicability of this topic to a satellite tasking application for the warfighter. Further application development needs input from the end users and to be tested for user-friendliness. Chapter II mentions that geocoding is supported in the Google Maps API. Geocoding should be integrated into the application.

3. Integration with Existing Programs for Access and Dissemination of Imagery

In Section C of Chapter I, there is a brief discussion of the existing NGA programs that provide access and dissemination of imagery. These programs were discussed for their possible integration with a tablet application. Further research into these programs is needed to fully assess the prospect of providing a tablet user interface into these systems.

4. Process Automation and Test Bed Application

This proof of concept worked from request to received data only with some human intervention. The request data needed to be synched with the MATLAB directory on the laptop. This was done with a FTP program called “Cyberduck.” This process could probably be easily automated but did not need to be for this thesis. Additional intervention was needed to synch the MATLAB output file with the server. Again, this process could probably be easily automated, but was not necessary for this project. When a tablet application is fully interfaced to a test bed there may no longer be a desire for complete automation. As in the operation of a real spacecraft system, a human-in-the-loop may be warranted to double check the commands before applying them to real hardware.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

A. LESSONS LEARNED

1. APPLE

The iOS, Apple’s Mobile Operating system, is updated frequently; check developer resources before updating software on the device used for development. Xcode needs to have the proper SDKs to build and install the application on the device used for development. Example: If Xcode 4.2.1 is installed on the Mac, and the iPad used for development has software version 5.0, everything may work fine. However, iTunes may be set to automatically download and update the iPad software. In this example situation, the iPad is updated to version 5.1.1. Xcode 4.2.1 is able to build the application, but cannot install the application on the iPad. The next version of Xcode may not be released for a while and the best solution is to restore the iPad back to version 5.0. System updates may take a long time to download and install. Alternatively, once the patch for the iOS simulator comes out for XCode, it can be downloaded via the App Store. Even if the App Store says the latest version is installed, check the XCode about information. The App Store will say that it is installed on the computer, but if you check XCode preferences it can be found in the Downloads tab and still needs to be installed in XCode.

2. MATLAB AND SIMULINK

For MATLAB 7.12.0 (R2011a) to properly compile SIMULINK in 64-bit Windows one needs to enter “mbuild –setup” in the command window and “mex –setup” in the command window. Microsoft Visual C++ 2010 Express ENU Setup, and Microsoft Windows SDK 7.1 for Windows Setup must be installed.

a. MATLAB Function Block Editor

Many issues can arise when using vectors in MATLAB function blocks within SIMULINK. One error states “Data 'variableName' (#83) is inferred as a variable size matrix, while its specified type is something else.” If the function block is opened and Edit Data/Ports is selected from the pull-down menu options for inheriting the data type or specifying the data type are available. Another way to adjust the size of a signal is to use the "Reshape" block available in SIMULINK.

b. The Reshape Block in SIMULINK

Coding in SIMULINK can create problems with vector sizing. A couple of "Reshape-Blocks" were used in order to force the data into the desired format for the next operation. An example of this block can be found in the CMG subsystem after the summation of $\mathbf{u}$ with the cross product of the integrated feedback and $\boldsymbol{\omega}$.

B. SIMULINK MODEL PROPERTIES INITFCN

```
% Simulation run parameters
SlewTime=6;

roll1=roll1*(pi/180); pitch1=pitch1*(pi/180); yaw1=yaw1*(pi/180);
%(radians)
roll2=roll2*(pi/180); pitch2=pitch2*(pi/180); yaw2=yaw2*(pi/180);
%(radians)
roll3=roll3*(pi/180); pitch3=pitch3*(pi/180); yaw3=yaw3*(pi/180);
%(radians)
roll4=roll4*(pi/180); pitch4=pitch4*(pi/180); yaw4=yaw4*(pi/180);
%(radians)
roll5=roll5*(pi/180); pitch5=pitch5*(pi/180); yaw5=yaw5*(pi/180);
%(radians)

%Constants
Re=6378e3;  % Earth Radius (m)
mu=398601.2e9; % Product of the gravitational constant and the mass of
earth (m^3/s^2)
we=0.000072921158553; % Earth's angular velocity rad/solar sec
(Vallado)


% Spacecraft orbit
h=681000;    % Orbit altitude meters
R=Re+h;    % Orbit radius from center of earth
wo=sqrt(mu/(Re+h)^3);  % Orbit angular velocity
incln=98*pi/180;   % GeoEye Inclination
epsilon=12*pi/180;
alphao=0;
uo=0; nuo=0;   % Start S/C beneath subsolar point
betasun=60; gamma=1.5;
a=0.545491852; b=0.314939867; c=0.704226952;
Area=[b*c a*c a*b];   % projected area~m^2 in body x,y,z directions
density=4.39e-14;
kpre=-9.9639/24/3600/180*pi*0; % nodal precession constant assumed zero
here
```

```
wn=kpre*(Re/(Re+h))^3.5*cos(incln); % nodal precession (zero
eccentricity)
V=wo*(Re+h);
rho=asin(Re/(h+Re));  % earth angular radius
Cd=2.5; psun=4.5E-6;  % Drag coefficient and solar pressure
constant~N/m^2
Kaero=-0.5*Cd*V^2; Psolar=2*psun; % constants for aero and solar torque
calculation
dL=[0.002 0.002 0.008];  % predicted distance between cp and cg
Kme=2.3390e-005;
mresid=[0 0 0.01];   % Spacecraft residual magnetic moment
M=mresid;    % Magnetic unit dipole vector
K=7.943e15;

%Spacecraft Inertia conditions
Imo=[119.1259 -15.7678 -6.5486;
  -15.7678 150.6615 22.3164;
  -6.5486 22.3164 106.0288];

Iinv=inv(Imo); % Moment of inertia inverse goes in dynamics block


%Spacecraft initial Euler state angles and rates
phio=0;thetao=0;psio=0;  %Initial Euler Angles
phidoto=0;thetadoto=0;psidoto=0; %Initial Euler Rates

%Calculation of initial quaternion (qo) and angular momentum (Ho)
s1=sin(phio/2);s2=sin(thetao/2);s3=sin(psio/2);c1=cos(phio/2);c2=cos(th
etao/2);c3=cos(psio/2);
q1o=s1*c2*c3-c1*s2*s3;
q2o=c1*s2*c3+s1*c2*s3;  %Wie pg. 321
q3o=c1*c2*s3-s1*s2*c3;
q4o=c1*c2*c3+s1*s2*s3;
S1=sin(phio);S2=sin(thetao);S3=sin(psio);C1=cos(phio);C2=cos(thetao);C3
=cos(psio);
wxo=phidoto-psidoto*S2-wo*S3*C2;
wyo=thetadoto*C1+psidoto*C2*S1-wo*(C3*C1+S3*S2*S1);
wzo=psidoto*C2*C1-thetadoto*S1-wo*(S3*S2*C1-C3*S1);
 qo=[q1o q2o q3o q4o];
 Ho=Imo*[wxo wyo wzo]';
 norm(Ho)*1000;


% CMG Properties (in degrees)
SaturationHi=pi; %was originall pi
SaturationLo=-pi;
beta=[54.73,54.73,54.73]; beta=beta.*pi/180; % Skew angle in degrees
Gimbal0=[-30;90;-30]*pi/180; % Initial Gimbal angles for 0 H spin-up
w_wheel=2800*(2*pi/60); % Wheel Speed in RPM Converted to rad/s
Iwheel=0.0614*1.3558179483314;% Wheel Inertia in slug-ft^2 Converted
(exact) to kg.m^2
h_wheel=Iwheel*w_wheel; % CMG Wheel Angular Momentum
```

C. LOADING APPLICATION REQUESTS INTO SIMULINK/MATLAB

```
% Courtney Guy
% Sample Maneuver

close all; clear all; clc;

Re=6378000; %m Earth Radius
h1=681000;  %m altitude of GeoEye-1 (make sure this matches Callbacks)
Ce=2*pi*Re; %m Earth Circumference

coord=importdata('StartA.txt', ',');
lat=coord(1);
long=coord(2);
targetLat0=lat;
targetLong0=long;

target=importdata('TargetB.txt', ',');
targetLat=target(1);
targetLong=target(2);

targetLatrad=targetLat*(pi/180);
latrad=lat*(pi/180);

deltaLat=(targetLat-lat)*(pi/180);
deltaLatMeters=deltaLat*Re;
deltaLong=(targetLong-long)*(pi/180)
deltaLongMeters=deltaLong*(Re*cos(pi*targetLat/180));

vara=sin(deltaLat/2)*sin(deltaLat/2)+cos(latrad)*cos(targetLatrad)*sin(
deltaLong/2)*sin(deltaLong/2);
varc=2*atan2(sqrt(vara), sqrt(1-vara));
distance=Re*varc;

reach=sqrt(distance^2+h1^2);

rollrad=atan(deltaLongMeters/h1);
rolldeg=rollrad*(180/pi);

pitchrad=atan(deltaLatMeters/h1);
pitchdeg=pitchrad*(180/pi)


roll1=rolldeg;
pitch1=pitchdeg;
yaw1=0;
targettime1=target(3);
targetLat1=targetLat;
targetLong1=targetLong;

lat=targetLat; % assign last target to new
long=targetLong;
```

```
% Import the next target and assign
target=importdata('TargetC.txt', ',');
targetLat=target(1);
targetLong=target(2);

targetLatrad=targetLat*(pi/180);
latrad=lat*(pi/180);

deltaLat=(targetLat-lat)*(pi/180);
deltaLatMeters=deltaLat*Re;
deltaLong=(targetLong-long)*(pi/180)
deltaLongMeters=deltaLong*(Re*cos(pi*targetLat/180));

vara=sin(deltaLat/2)*sin(deltaLat/2)+cos(latrad)*cos(targetLatrad)*sin(
deltaLong/2)*sin(deltaLong/2);
varc=2*atan2(sqrt(vara), sqrt(1-vara));
distance=Re*varc;

reach=sqrt(distance^2+h1^2);

rollrad=atan(deltaLongMeters/h1);
rolldeg=rollrad*(180/pi);

pitchrad=atan(deltaLatMeters/h1);
pitchdeg=pitchrad*(180/pi)


roll2=rolldeg;
pitch2=pitchdeg;
yaw2=0;
targettime2=target(3);
targetLat2=targetLat;
targetLong2=targetLong;

lat=targetLat; % assign last target to new
long=targetLong;


target=importdata('TargetD.txt', ',');
targetLat=target(1);
targetLong=target(2);

targetLatrad=targetLat*(pi/180);
latrad=lat*(pi/180);

deltaLat=(targetLat-lat)*(pi/180);
deltaLatMeters=deltaLat*Re;
deltaLong=(targetLong-long)*(pi/180)
deltaLongMeters=deltaLong*(Re*cos(pi*targetLat/180));
```

```
vara=sin(deltaLat/2)*sin(deltaLat/2)+cos(latrad)*cos(targetLatrad)*sin(
deltaLong/2)*sin(deltaLong/2);
varc=2*atan2(sqrt(vara), sqrt(1-vara));
distance=Re*varc;

reach=sqrt(distance^2+h1^2);

rollrad=atan(deltaLongMeters/h1);
rolldeg=rollrad*(180/pi);

pitchrad=atan(deltaLatMeters/h1);
pitchdeg=pitchrad*(180/pi)


roll3=rolldeg;
pitch3=pitchdeg;
yaw3=0;
targettime3=target(3);
targetLat3=targetLat;
targetLong3=targetLong;

lat=targetLat; % assign last target to new
long=targetLong;

% Import the next target and assign
target=importdata('TargetE.txt', ',');
targetLat=target(1);
targetLong=target(2);

targetLatrad=targetLat*(pi/180);
latrad=lat*(pi/180);

deltaLat=(targetLat-lat)*(pi/180);
deltaLatMeters=deltaLat*Re;
deltaLong=(targetLong-long)*(pi/180)
deltaLongMeters=deltaLong*(Re*cos(pi*targetLat/180));

vara=sin(deltaLat/2)*sin(deltaLat/2)+cos(latrad)*cos(targetLatrad)*sin(
deltaLong/2)*sin(deltaLong/2);
varc=2*atan2(sqrt(vara), sqrt(1-vara));
distance=Re*varc;

reach=sqrt(distance^2+h1^2);

rollrad=atan(deltaLongMeters/h1);
rolldeg=rollrad*(180/pi);

pitchrad=atan(deltaLatMeters/h1);
pitchdeg=pitchrad*(180/pi)


roll4=rolldeg;
```

```
pitch4=pitchdeg;
yaw4=0;
targettime4=target(3)
targetLat4=targetLat;
targetLong4=targetLong;

lat=targetLat; % assign last target to new
long=targetLong;

% Return to Start
target=importdata('StartA.txt', ',');
targetLat=target(1);
targetLong=target(2);

targetLatrad=targetLat*(pi/180);
latrad=lat*(pi/180);

deltaLat=(targetLat-lat)*(pi/180);
deltaLatMeters=deltaLat*Re;
deltaLong=(targetLong-long)*(pi/180)
deltaLongMeters=deltaLong*(Re*cos(pi*targetLat/180));

vara=sin(deltaLat/2)*sin(deltaLat/2)+cos(latrad)*cos(targetLatrad)*sin(
deltaLong/2)*sin(deltaLong/2);
varc=2*atan2(sqrt(vara), sqrt(1-vara));
distance=Re*varc;

reach=sqrt(distance^2+h1^2);

rollrad=atan(deltaLongMeters/h1);
rolldeg=rollrad*(180/pi);

pitchrad=atan(deltaLatMeters/h1);
pitchdeg=pitchrad*(180/pi)


roll5=rolldeg;
pitch5=pitchdeg;
yaw5=0;
targettime5=targettime4+6+10;
targetLat5=targetLat;
targetLong5=targetLong;

lat=targetLat; % assign last target to new
long=targetLong;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rpyt=[roll1, roll2, roll3, roll4, roll5;  pitch1 pitch2, pitch3, ...
    pitch4, pitch5;  yaw1, yaw2, yaw3, yaw4, yaw5; targettime1, ...
    targettime2, targettime3, targettime4, targettime5];

rpyt(1:3,5) = -sum(rpyt(1:3,1:4),2);
%Corrects "rounding error"
```

```
rpytc=[0,roll1, roll2, roll3, roll4, roll5;  0,pitch1 pitch2, pitch3,
...
    pitch4, pitch5;  0, yaw1, yaw2, yaw3, yaw4, yaw5; 0, targettime1,
...
    targettime2, targettime3, targettime4, targettime5;
    targetLat0, targetLat1,targetLat2,targetLat3,targetLat4,targetLat5;
    targetLong0,
targetLong1,targetLong2,targetLong3,targetLong4,targetLong5];
rpytc(1:3,6) = rpyt(1:3,5);
rpytc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

D. POSITION, VELOCITY AND ACCELERATION TRAJECTORY

```
function [tVec, wVec, w_dotVec] = fcn(tVecOld, time, rpy, ~, index)

wVec=zeros(3,1);
w_dotVec=zeros(3,1);
tVec=zeros(3,1);
constant=zeros(3,1);

if time < rpy(4, 1)
 tVec=[0;0;0];
 wVec=[0;0;0];
 w_dotVec=[0;0;0];
end

if time >= rpy(4, index) && time < (rpy(4, index)+2)
 if index > 1
 for index2=1:1:index-1;
 constant=(rpy(1:3, index-index2))+constant;
 end
 end
 wmax=(rpy(1:3, index))./4;
 tVec=(wmax./4).*(time-rpy(4, index)).^2+constant;
 wVec=(wmax./2).*(time-rpy(4, index));
 w_dotVec=wmax./2;
 if wmax(1)==0
 tVec(1)=constant(1);
 end
 if wmax(2)==0
 tVec(2)=constant(2);
 end
end

if time >= (rpy(4, index)+2) && time < (rpy(4, index)+4)
 if index > 1
 for index2=1:1:index-1
 constant=(rpy(1:3, index-index2))+constant;
 %constant=(rpy(1:3, index-index2))./2+constant;
 end
 end
 wmax=(rpy(1:3, index))./4;
% tVec=2.*wmax.*(time-(rpy(4, index)+2))+constant;
 tVec=wmax.*(time-(rpy(4, index)+2))+wmax+constant;
 wVec=wmax;
 w_dotVec=[0;0;0];
 if wmax(1)==0
 tVec(1)=constant(1);
 end
 if wmax(2)==0
 tVec(2)=constant(2);
 end
end
```

```
if time >= (rpy(4, index)+4) && time < (rpy(4, index)+6)
 if index > 1
 for index2=1:1:index-1
 constant=(rpy(1:3, index-index2))+constant;
 end
 end
 wmax=(rpy(1:3, index))./4;
% tVec=(wmax./2).*(time-(rpy(4, index)+4)).^2+2*wmax+constant; % This
% one works but is not the integral of the velocity vector.
% tVec=wmax.*(2*(time-(rpy(4, index)+4)))-(((wmax.*(time-(rpy(4,
index)+4))).^2)/2)+constant;
% tVec=wmax.*(2*(time-(rpy(4, index)+4)))-(((wmax.*(time-(rpy(4,
index)+4))).^2)/2)+2.*wmax+constant;
% tVec=wmax.*(2*(time-(rpy(4, index)+4)))-(((wmax.*(time-(rpy(4,
index)+4))).^2)/2)+2.*wmax
% tVec=2.*(time-(rpy(4, index)+4)).*wmax-((((time-(rpy(4,
index)+4)).^2).*wmax)./(4))+2.*wmax+constant;
 constant2=3.*wmax;
 tVec=(time-(rpy(4, index)+4)).*wmax-((((time-(rpy(4,
index)+4)).^2).*wmax)./(4))+constant+constant2;
 wVec=-(wmax./2).*(time-(rpy(4, index)+4))+wmax;
 w_dotVec=-wmax./2;
 if wmax(1)==0
 tVec(1)=constant(1);
 end
 if wmax(2)==0
 tVec(2)=constant(2);
 end
end

if index >1 && time < rpy(4, index)
 for index2=1:1:index-1
 constant=(rpy(1:3, index-index2))+constant;
 end
 tVec=constant;
end
```

LIST OF REFERENCES

Access Intelligence LLC. 2010. “DigitalGlobe Issues WorldView-3 Development Deals to Ball Aerospace, ITT.” *Satellite Today*, September 2.

———. 2011. “NGA Expands DigitalGlobe EnhancedView Contract.” *Satellite Today*, October 06.

Air Land Sea Application Center. 2004. *UHF TACSAT/DAMA.* Multi-Service Tactics Techniques and Procedures, Fort Monroe, Virgina: U.S. Army Training and Doctrine Command.

Apple. 2012. *Developer–iOS Provisioning Portal.* September 7. https://developer.apple.com/ios/my/provision/index.action.

Apple Inc. 2012. “iOS Developer Library.” *iOS Human Interface Guidelines.* August 31. http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html#//apple_ref/doc/uid/TP40006556.

Bate, Roger R., Donald D. Mueller, and Jerry E. White. 1971. *Fundamentals of Astrodynamics.* Mineola, NY: Dover Publications.

Bergin, Chris. 2012. *NASA Spaceflight.* August 9, 2012. http://www.nasaspaceflight.com/2012/08/usaf-kestrel-eye-1-spacecraft-falcon-9-2013.

Birk, Ronald J, Thomas Stanley, Gregory I. Snyder, Thomas A. Hennig, Matthew M. Fladeland, and Fritz Policelli. 2003. “Government programs for research and operational uses of commercial remote sensing data.” *Remote Sensing of Environment* 88: 3–16.

Blocker, Allen, Chance Litton, Jason Hall, and Marcello Romano. 2008. “TINYSCOPE - The Feasibility of 3-Axis Stabilized Earth Imaging CubeSat from LEO.”

Crampton, Jeremy W., Susan M. Roberts, and Ate Poorthuis. 2011.”The New Political Economy of Geographical Intelligence.” *Annals of the Association of American Geographers*. 196–21.

Cutshaw, Jason B. 2012. *EVR2EST helps firefighters during wildfires.* Redstone Arsenal, Ala.

Cutshaw, Jason B. 2012. *SMDC employee helps nation prepare for emergencies.* Redstone Arsenal, Ala.

Defense Daily. 1999. “Commercial imagery to provide commanders rapid pictures.” 1.

DigitalGlobe. 2015. “QuickBird Data Sheet.” *DigitalGlobe.* June 1. http://www.digitalglobe.com.

———. 2015. *Satellite Imagery and Geospatial Information Products.* March 23. www.digitalglobe.com.

Driels, M. 1996. *Linear Control Systems Engineering.* San Francisco: McGraw-Hill.

Eisler, Peter. 2008. “Google Earth Helps Yet Worries Government.” *USA Today*, November 7.

Engvall, Christian. 2012 *Christian Engvall Blog.* http://www.christianengvall.se/phonegap-and-google-maps.

Frakes, Dan. 2012. *Macworld.* August 5. http://www.macworld.com/article/1142125/iphonedevcamp3.

GeoEye. 2012 “GeoEye.” *GeoEye.* March 1. http://www.geoeye.com/CorpSite/assets/docs/brochures/GeoEye-1_Fact_Sheet.pdf.

Gildea, Kerry. 2002. “Lawmakers put pressure on DoD to devise commercial imagery strategy.” June 6. *C4I News*.

Gillett, Frank. 2012. “Why Tablets Will Become Our Primary Computing Device.” *Forrestor Blog.* April 23. Cambridge, Massachusetts.

Globalstar. 2011. “GSP-1720 satellite data and voice module.” *Globalstar.* August 29. http://www.globalstar.com/shop/index.php?main_page=product_info&cPath=23&products_id=81.

Google. 2012. *Android Developer Tools | Android Developers.* September 9. http://developer.android.com/tools/help/adt.html.

———. 2012. *Google Earth projection - Google Earth Help*. June 2. http://support.google.com/earth/bin/answer.py?hl=en&answer=148110.

Hartmetz, James A. 2001. “Eagle Vision - Exploiting Commercial Satellite Imagery.” *The DISAM Journal*, 22–25.

Hinckley, Ken, Michel Pahud, and Bill Buston. 2010 “Direct Display Interaction via Simultaneous Pen and Multi-touch Input.” *Society for Information Display (SID) Symposium Digest of Technical Papers* 41, no. 1: 537–540.

Hodgson, Micheal E., and Bandana Kar. 2008. “Modeling the Potential Swath Coverage of Nadir and Off-Nadir Pointable Remote Sensing Satellite-Sensor Systems.” *Cartography and Geographic Information Science* 35, no. 3: 147–156.

Keller, John. 2012. “DARPA Seeks to Develop Small Reconnaissance Satellites That are Cheaper to Build than UAVs.” *Military and Aerospace Electronics*. June: 6–8.

KMI Media Group. 2009 “Kestrel Eye Update.” *Military Space & Missile Forum* 2, no. 5.

Krebs, Gunter. 2012 *Gunter's Space Page.* June 10. http://space.skyrocket.de/doc_sdat/orbview-3.htm.

Kukulska-Hulme, Agnes, and John Traxler. 2005. *Mobile Learning: A Handbook For Educators and Trainers.* New York: Taylor and Francis, Inc.

Lambeth, Benjamin. 2001. *Air Power Against Terror: America's Conduct of Operation Enduring Freedom.* Santa Monica: RAND Corporation.

LeRoux, Brian. 2012. *PhoneGap Blog.* March 19. http://phonegap.com/2012/03/19/phonegap-cordova-and-what%E2%80%99s-in-a-name.

Litton, C. Chance. 2009. “TINYSCOPE: The Feasiblity of a Tactically Useful Earth-Imaging Nanosatellite and a Preliminary Design of the Optical Payload.” *Master's thesis.* Monterey, CA: Naval Postgraduate School.

Maathuisa, B. H. P., and J.L. van Genderena. 2004. “A review of satellite and airborne sensors for remote sensing based detection of minefields and landmines.” *International Journal of Remote Sensing*. 5201–5245.

McLaughlin, Col. K. 2007. “Operationaly Responsive Space Office - Director's Brief.” *Responsive Space.* July. http://www.responsivespace.com/ors/reference/McLaughlin.pdf.

Mian, Salman, Jose Teixeira, and Eija Koskivaara. 2011. “Open-Source Software Implications in the Competitive Mobile Platforms Market.” *Building the e-World Ecosystem*, 110-128. Boston: Springer.

Miller, Steve. 2011. “GeoEye Presentation to The National Guard Space & Missile Defense Symposium.” *Fourth Annual National Guard Space & Missile Defense Symposium.* Colorado Springs: Colorade National Guard. 1–40.

Molich, Rolf, and Jakob Nielsen. 1990. “Improving a Human-Computer Dialogue.” *Communications of the ACM* 33, no. 3: 338-348.

National Security Space Office. 2007. *Plan for Operationally Responsive Space.* Report to Congressional Defense, Washington, DC: Department of Defense.

Oetting, John D., and Tao Jen. 2011. “The Mobile User Objective System.” *Johns Hopkins APL Technical Digest*: 103–112.

Office of Geospatial-Intelligence Management. 2006. *Geospatial Intelligence (GEOINT) Basic Doctrine.* Doctrine, Bethesda, MD: National Geospatial-Intelligence Agency.

Oulasvirta, Antti, and Joanna Bergstrom-Lehtovirta. 2011. "Ease of Juggling: Studying the Effects of Manual Multitasking." *Proceedings of the 2011 Annual Conference on Human factors in Computing Systems.* New York, NY: ACM. 3103–3112.

Raytheon. 2015. *Raytheon.* February 4. http://www.raytheon.com/news/feature/rms13_seeme.html.

Robusto, C. C. 1957. "The Cosine-Haversine Formula." *The American Mathematical Monthly* 64, no. 1: 38–40.

Sands, Timothy. 2012. "Lecture 7." February 14. *AE4901 Spacecraft Attitude DYnamics and Control.* Monterey, CA: Sakai CLE.

Satellite Imaging Corporation. 2015. *IKONOS Satellite Sensor.* January 22. http://www.satimagingcorp.com/satellite-sensors/.

Sencha Inc. 2012. "Building your First App." *Sencha Incorporated.* August 14. http://docs.sencha.com/touch/2-0/#!/guide/first_app.

Shankland, Shankland. 2008. *Google to buy GeoEye satellite imagery.* August 29. http://news.cnet.com/8301-1023_3-10028842-93.html?part=rss&subj=news&tag=2547-1023_3-0-5.

Sidi, Marcel J. 2006. *Spacecraft Dynamics & Control.* New York: Cambridge University Press.

USASMDC/ARSTRAT/Public Affairs Office. 2010. "Kestrel Eye Fact Sheet." *U.S. Army Space & Missile Defense Command.* February 2. http://www.smdc.army.mil/FactSheets/KestrelEye.pdf.

Vance, Leonard, Rigel Woida, and Chad Spalt. 2014. "Adaptation of Manufacturing to Mass Production of Nanosatellites." *Small Satellite Conference.* Utah.

ViaSat. 2012. "ViaSat." *SurfBeam 2 Pro Portable Terminal.* https://www.viasat.com/sites/default/files/legacy/survbeam2_ProPortable_Datasheet_019_web.

Wasserman, Anthony I. 2010. "Software engineering issues for mobile application development." *FoSER '10 Proceedings of the FSE/SDP workshop on Future of software engineering research.* New York: ACM. 397–400.

Wei, Bong. 2008. “Rotational Maneuvers and Attitude Control.” In *Space Vehicle Dynamics and Control Second Edition*, by Bong Wei, 403-482. Blacksburg, VA: American Institute of Aeronautics and Astronautics, Inc.

Weisstein, Eric W. 2012. *Hermitian Matrix.* March 5. http://mathworld.wolfram.com/HermitianMatrix.html.

Wentk, Richard. 2011. *Xcode 4.* Indianapolis: Wiley Publishing, Inc.

Zhon, Jinghua. 2012. “PID Controller Tuning, a Short Tutorial.” June. http://wwwdsa.uqac.ca/~rbeguena/Systemes_Asservis/PID.

Zipfel, Peter H. 2000. *Modeling and Simulatio of Aerospace Vehicle Dynamics.* Reston: American Institute of Aeronautics and Astronautics, Inc.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California